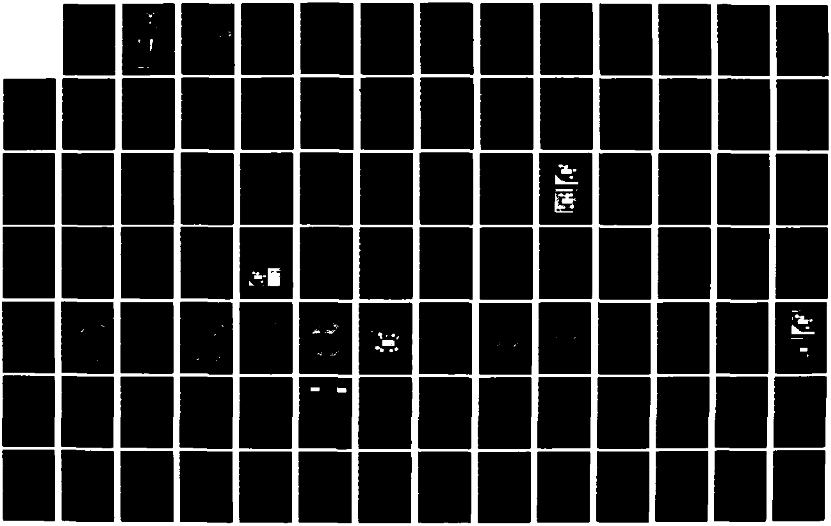
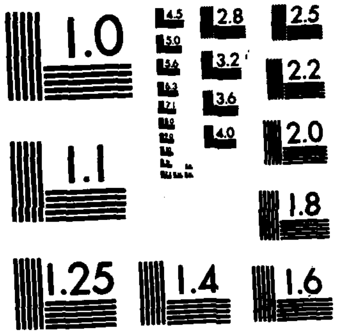
AD-A124 788   SCENE ANALYSIS: NON-LINEAR SPATIAL FILTERING FOR   1/2
AUTOMATIC TARGET DETECTION(U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI..   J H CROMER

UNCLASSIFIED   DEC 82 AFIT/GE/EE/82D-26   F/G 17/8   NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

UNITED STATES AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/EE/82D-26

1

SCENE ANALYSIS: NON-LINEAR SPATIAL FILTERING FOR AUTOMATIC TARGET DETECTION

THESIS

AFIT/GE/EE/82D-26 JAMES H. CROMER
2nd Lt USAF

Approved for public release; distribution unlimited.

AFIT/GE/EE/82D-26

SCENE ANALYSIS: NON-LINEAR SPATIAL FILTERING FOR AUTOMATIC TARGET DETECTION

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by

James H. Cromer, B.S.
2nd Lt USAF

Graduate Electrical Engineering

December 1982

Approved for public release; distribution unlimited.

Preface

One purpose of this study was to investigate a clutter-energy invariant target detection algorithm. A secondary purpose was to develop a base of image processing software for the AFIT Digital Signal Processing Laboratory NOVA-ECLIPSE minicomputer system, for use by future thesis students.

I would like to thank my advisor, Dr. Matthew Kabrisky, and committee members Major Larry Kizer and Major Kenneth Castor for their time and assistance during this study. I especially wish to express my gratitude to my wife, Karen, for her ceaseless support during my time at AFIT.

James H. Cromer

Contents

List of Figures

List of Tables

List of Symbols

$t(x,y)$	--	array representing the template
$s(x,y)$	--	array representing the scene
$\triangleq$	--	"is defined as"
$\sum(\bullet)$	--	summation of $(\bullet)$
$x * y$	--	region of overlap of scene and shifted template
$\lvert(\bullet)\rvert$	--	absolute value of $(\bullet)$
D1(m,n)	--	distance metric based on L1 norm
D2(m,n)	--	distance metric based on L2 norm
t	--	"the vector t"
$(\bullet)^T$	--	vector transpose of $(\bullet)$
Es(m,n)	--	energy of scene search window centered at (m,n)
Et	--	energy of template
Rst(m,n)	--	cross-correlation between scene and template
Nst(m,n)	--	normalized cross-correlation
NL1(m,n)	--	normalized L1 distance function
NL2(m,n)	--	normalized L2 distance function
SL1E	--	L1 energy of scene
TL1E	--	L1 energy of template
CFACTOR	--	correlation distance factor
L2FACTOR	--	L2 distance factor
L1FACTOR	--	L1 distance factor
$\tilde{g}(x,y)$	--	periodic extension of g(x,y)
$\longleftrightarrow$	--	indicates Fourier Transform pair
$F\{(\bullet)\}$	--	Fourier Transform $(\bullet)$

$F^{-1}\{(\bullet)\}$	--	inverse Fourier Transform of $(\bullet)$
T^*	--	complex conjugate of T
(Hi,Vj)	--	grid rectangle in grid row j, grid column i
s'(x,y)	--	normalized array
Nij	--	normalized coefficient
Eij	--	energy of grid rectangle (Hi,Vj)
Ng(m,n)	--	statistical correlation measure

ABSTRACT

This work focuses on a method for two-dimensional pattern recognition. The method includes a global search scheme for candidate windows of interest, based on Fourier domain cross-correlation. A method to normalize the input scene by local rectangular regions, in an attempt to efficiently approximate search window normalization, is presented. Also developed is a candidate window (potential target) similarity measure, based on the normalized L1 and Euclidean distances, which is independent of the template DC value and its energy. Observations on the performance of the algorithm applied to visual spectrum photographs of tanks in a realistic environment are included. Also included is the software needed to implement the algorithm on a Data General Eclipse S/250 minicomputer.

SCENE ANALYSIS: NON-LINEAR SPATIAL FILTERING FOR AUTOMATIC TARGET DETECTION

I. INTRODUCTION

GENERAL

Target detection is the area of pattern recognition concerned with locating a given class of objects embedded within a background scene. The distinction to be made is whether the object in question belongs in the target class or in the non-target class. The challenge of this type of automatic analysis of complex visual data by machine has proven to be surprisingly difficult. The problem remains largely unsolved despite considerable research effort [1: 28].

JUSTIFICATION

Automated systems capable of identifying objects in a cluttered background, irrespective of size, orientation, illumination, or position would have a nearly unlimited range of applications. The following list suggests a few of the major areas [2: 596]:

1)Document processing: recognition of unformatted, non-segmented, multi-font characters;

2)Industrial automation: robot assembly and inspection;

3)Military applications: analysis of reconnaissance imagery, enhancement of weapon delivery display systems, and realization of the autonomous missile.

BACKGROUND

The target detection process can be broken down into three steps as follows:

1)Determine characteristic features of the target template that are invariant to size, orientation, energy, and background changes;

2)Perform a global search of the cluttered input scene for the features of step (1) and identify local "windows" of interest (target candidates); and

3)Classify the information within the candidate windows into one of the two disjoint classes of either targets or non-targets.

The problems associated with this target detection process are numerous. One problem is in analytically describing the quintessence of a class of objects, given the physical characteristics of only a finite number of template objects. Once a suitable set of target characteristics (or features) is selected, the isolation of those features from a background scene which may posess some of the same features as the target may be impossible. For these and other reasons, no general solution exists to detect objects embedded in "real-life" cluttered scenes.

Previous studies performed in this area at the Air Force Institute of Technology have shown promising, although limited, results. One of these studies is a thesis by

Israeli Air Force Major Moshe Horev, "Picture Correlation Model for Automatic Machine Recognition" [3]. In it, Horev describes a series of transformations performed in the Fourier domain to determine the size and orientation of targets within a cluttered scene. He then suggests to perform a non-linear (and, hence potentially unpredictable) operation of combining the modified phase of the scene with the magnitude of the template in hopes of enhancing the target objects; this procedure is known as the "Phase of the Image, Magnitude of the Template" (PIMT) process. An immediate observation is that the success of the process may be both scene and template dependent. The PIMT process has debatable merit, but the existence of a scale-rotation transformation does suggest an area for further research. With the premise that the size and orientation of a candidate target within a scene is known or can be determined, can a process be developed to then locate and accurately classify the target?

PROBLEM/SCOPE

This study includes:

1)The initial development of a target detection algorithm that is invariant to background scene composition or energy;

2)The implementation on a digital minicomputer of the software routines needed to process the input imagery, perform global scene searches through high-speed correlation, and discriminate local candidate windows by using L1 and L2 distance metrics; and

3)The preliminary test results. A complete statistical performance analysis of the process was not conducted due to time constraints. Suggestions for improving the performance of the process are included.

ASSUMPTIONS

The following assumptions were made concerning the input test scenes:

1)The size and orientation of targets in a scene is known, or can be determined by existing methods, three of which are diffraction pattern sampling, cross-correlation with a bank of scaled and rotated templates, or by performing a scale-rotation transformation;

2)The digitized scene images are accurate representations of the continuous scenes from which they were obtained;

3)The digitized scene images may have been corrupted by additive uncorrelated noise; and

4)Illumination over the continuous scene is varying slowly.

OVERVIEW OF PRESENTATION

The next chapter discusses methods of detecting objects in scenes through the classic technique of template matching. Distance factors based on the L1 and L2 distance metrics are derived; these factors will be used to classify candidate windows, or potential targets.

Chapter three briefly discusses the discrete Fourier transform and some of its properties. A method for performing high-speed correlation by multiplication in the Fourier domain is given; cross-correlation will be used in the detection process to perform searches for potential targets in the input scene.

In chapter four a scene normalization scheme is presented. The normalization is necessary to improve the ability of the cross-correlation to locate candidate targets.

The software needed to implement the detection process is described in chapter five. The source code has been included in the appendix.

In chapter six, some observations of detection process are made. Explanations are given for the weaknesses of the algorithm, and suggestions for improving the performance are discussed.

II. TEMPLATE MATCHING

Often in scene analysis problems a simple question is to be answered: Does the input scene contain a previously specified object? A technique classically employed to determine the presence of an object is the fundamental method of template matching, in which the template brightness function is compared point-by-point with the scene brightness function. In most cases, a perfect template match will not be found, so some realistic distance measure D(m,n) indicating the degree of similarity between the template window and the scene needs to be computed for all possible points in the scene.

L1 and L2 DISTANCE MEASURES

Let the array (or vector) t(x,y) represent in some sense the template pattern, and let the array s(x,y) represent the scene to be searched. For our purposes of discussion, it is immaterial how the arrays are obtained, whether from digitizing (sampling and quantizing) the continuous brightness distributions, infrared distributions, or some function of these distributions (for example only the low-frequency Fourier components). Two common definitions of distance measures used are given by equations (1) and (3) [4: 279].

$$D1(m,n) \triangleq \sum_{x} {}_{*} \sum_{y} |s(x,y) - t(x-m,y-n)| \qquad (1)$$

$$= \sum_{x} {}_{*} \sum_{y} \sqrt{[s(x,y) - t(x-m,y-n)]^{2}} \qquad (2)$$

$$D2(m,n) \triangleq \sqrt{\sum_{x} {}_{*} \sum_{y} [s(x,y) - t(x-m,y-n)]^{2}} \qquad (3)$$

* -- for all x,y such that (x-m,y-n) is within the area of overlap of the scene and template windows (0 < m < M+J, 0 < n < N+K for a JxK template and an MxN search area). See Figure 1 for an illustration of the labelling convention used. Note that m and n represent a specific translation between s(x,y) and t(x,y).

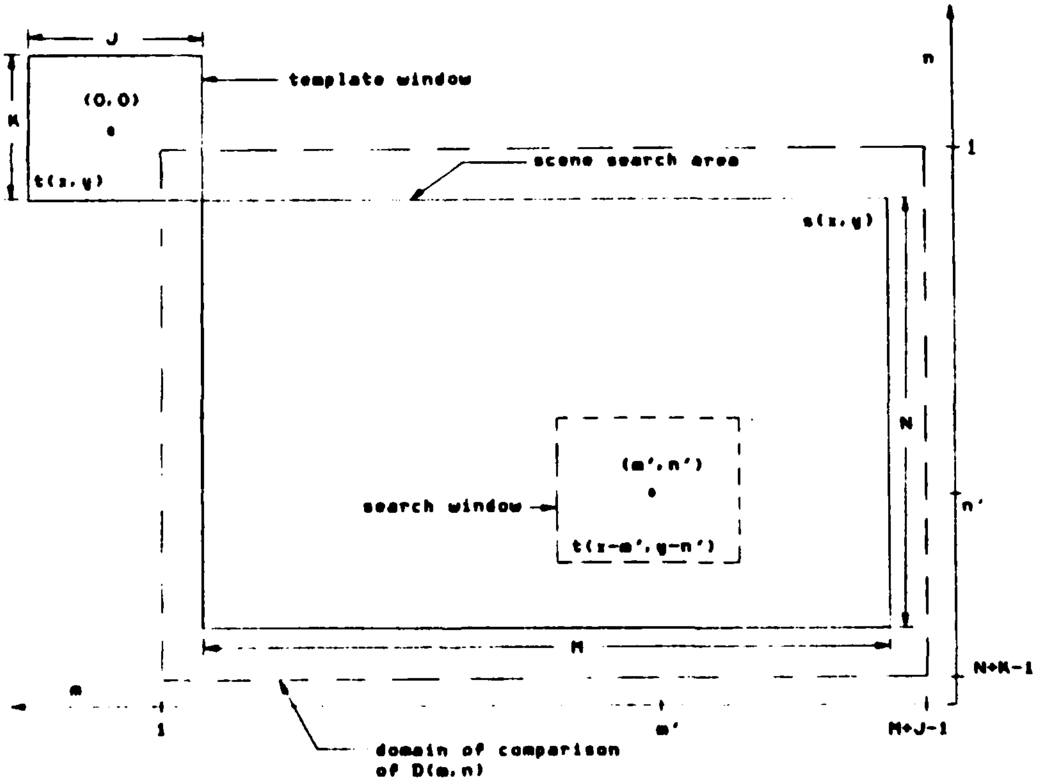


FIGURE 1 SEARCH WINDOW LABELLING SCHEME FOR COMPUTATION OF DISTANCE MEASURE D(m',n')

The definitions for D1(m,n) and D2(m,n) are known as the metrics based on the L1 and L2 norms respectively. The D2(m,n) measure is also known as the standard Euclidean distance between two vectors, that is

$$D2(m,n) = (s - \hat{t})^{T} (s - \hat{t}) \qquad (4)$$

$$\text{where} \quad \hat{t} = \begin{bmatrix} t(x1-m,y1-n) \\ t(x2-m,y2-n) \\ \vdots \\ t(xn-m,yn-n) \end{bmatrix}$$

NORMALIZED CROSS-CORRELATION

Insight can be gained from equation (3) by expanding it as follows:

$$D2(m,n)^{2} = \sum_{x}{}_{*}\sum_{y} \Big[s^{2}(x,y) - 2s(x,y)t(x-m,y-n) + t^{2}(x-m,y-n) \Big] \qquad (5)$$

Equivalently,

$$D2(m,n)^{2} = \sum_{x}{}_{*}\sum_{y} s^{2}(x,y) - 2\sum_{x}\sum_{y} s(x,y)t(x-m,y-n) + \sum_{x}\sum_{y} t^{2}(x-m,y-n) \qquad (6)$$

Let Es, Et, and Rst be defined as follows:

$$Es(m,n) \triangleq \sum_{x}{}_{*}\sum_{y} s^{2}(x,y) \qquad (7)$$

$$Et(m,n) \triangleq \sum_{x}\sum_{y} t^{2}(x-m,y-n) \qquad (8)$$

$$Rst(m,n) \triangleq \sum_{x}\sum_{y} s(x,y)t(x-m,y-n) \tag{9}$$

(There is no restriction on the range of x and y in equations (8) and (9) as the template function is considered to be zero outside the area of interest.)

The term Es(m,n) represents the scene energy (or equivalently the vector length) of the search window, which will vary cver the search area. The term Et(m,n) represents the template energy, which is constant for all values of m and n. The term Rst(m,n) is the cross-correlation between the scene and t template, and generally is largest when the distance D2(m,n) i- smallest. The cross-correlation term is not an absolute measure of the template difference, however, since the scene window energy Es(m,n) is position variant. For this reason, Rst(m,n) is normalized to achieve invariance to input energy [5: 553]. The normalized cross-correlation, denoted by Nst(m,n), is defined as follows:

$$Nst(m,n) \triangleq \frac{Rst(m,n)}{\sqrt{Et(m,n)}\,\sqrt{Es(m,n)}} \tag{10}$$

where the usual restriction is placed on x and y for the computation of the scene energy term.

The significance of this normalization can be realized by appealing to the Schwarz inequality [6: 159]. The Schwarz inequality is stated as follows:

For two real functions f(x) and g(x) defined on a<x<b,

$$\sum f(x)g(x) \leq \sqrt{\sum [f(x)]^{2}} \sqrt{\sum [g(x)]^{2}} \qquad (11)$$

with equality when f(x)=kg(x), where k is a constant scale factor. As applied to the cross-correlation terms,

$$\sum_{x} \sum_{y} s(x,y)t(x-m,y-n) \leq \sqrt{[\sum_{x} * \sum_{y} s^{2}(x,y)] \quad [\sum_{x} \sum_{y} t^{2}(x-m,y-n)]} \qquad (12)$$

or, by rearranging terms

$$Nst(m,n) \leq 1 \qquad (13)$$

with equality only when the scene area under consideration exactly matches the template. Thus the normalized cross-correlation can be used as a decision criterion regardless of the distribution of the non-normalized scene energy, assuming that the scene energy can be recomputed for each shift of the search window. A simple decision rule would classify the scene window information into the target class only when Nst exceeded some preset upper threshold value, into the non-target class when Nst was below a lower threshold, and would not make a decision when Nst was between the thresholds.

With the constraint that the scene and template can take on only non-negative values, a lower bound for the threshold value is zero. Fortunately, a tighter bound closer to unity can be determined for cross-correlations with scenes of interest (ones that approach the template in form). Consider a scene of constant value c, c>0. Then the cross-correlation value may be determined by applying equations (7) through (10):

$$Es = \sum_{x}\sum_{y} c^{2} \tag{14}$$

$$Es = c^{2} JK \tag{15}$$

$$[Nst|s=c] = \frac{c \sum_{x} \sum_{y} t(x-m,y-n)}{c \sqrt{JK} \sqrt{Et}} \tag{16}$$

$$Nst|c = \frac{\sum_{x} \sum_{y} t(x-m,y-n)}{\sqrt{EtJK}} \tag{17}$$

with J = width of search window
K = length of search window

Note that the normalized cross-correlation between the template and a scene of constant non-zero value, Nst|c, is independent of the scene value. Scene windows which yield a Nst less than Nst|c need not be considered for further discrimination.

NORMALIZED L1 and L2 DISTANCES

Two other distance measures which take into account the array energies are the normalized L1 and L2 distance measures, defined in Eqs. (18) and (19).

$$\mathrm{NL1}(m,n) \triangleq \sum_{x} * \sum_{y} \left| \frac{s(x,y)}{\mathrm{SL1E}} - \frac{t(x-m,y-n)}{\mathrm{TL1E}} \right| \qquad (18)$$

$$\mathrm{NL2}(m,n) \triangleq \sqrt{\sum_{x} * \sum_{y} \left(\frac{s(x,y)}{\mathrm{Es}(m,n)} - \frac{t(x-m,y-n)}{\mathrm{Et}} \right)^2} \qquad (19)$$

with

$$\mathrm{SL1E} \triangleq \sum_{x} * \sum_{y} s(x,y) \qquad (20)$$

$$\mathrm{TL1E} \triangleq \sum_{x} \sum_{y} t(x-m,y-n) \qquad (21)$$

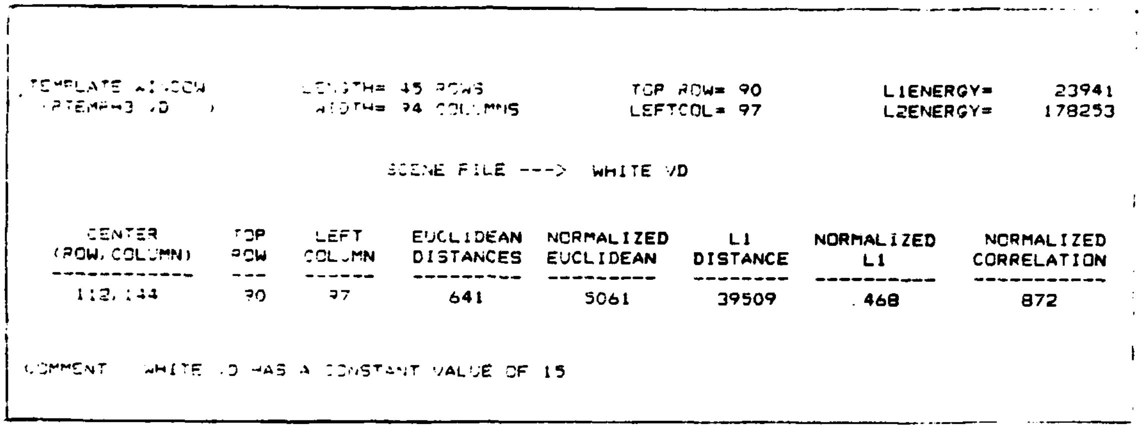
The normalized L2 distance can be determined more efficiently by Eq. (22):

$$\mathrm{NL2}(m,n) = \sqrt{2\ [1 - \mathrm{Nst}(m,n)]} \qquad (22)$$

One way of "visualizing" the normalized L2 distance is to think of it as the Euclidean distance between the points where the scene vector and the template vector intersect the unit hypersphere. Thus NL2 is dependent only upon the angle between the vectors, and not on the vector lengths.

The maximum normalized distances to be considered as possibly identifying a target location will be those corresponding to the distances computed between a template and a scene with a constant value. These distances are given in Table I. for typical tank template.

TABLE I. DISTANCES FROM TEMPLATE H3 TO A CONSTANT VALUED SCENE



```
TEMPLATE WINDOW          LENGTH= 45 ROWS         TOP ROW= 90        L1ENERGY=     23941
 (PTEMPH3 .VD    )        WIDTH= 94 COLUMNS      LEFTCOL= 97        L2ENERGY=    178253

                             SCENE FILE --->  WHITE .VD

    CENTER        TOP    LEFT    EUCLIDEAN  NORMALIZED    L1      NORMALIZED    NORMALIZED
 (ROW,COLUMN)     ROW   COLUMN   DISTANCES  EUCLIDEAN  DISTANCE       L1       CORRELATION
 ------------     ---   ------   ---------  ---------  --------   ----------   -----------
   112,144         90     97        641       5061      39509        .468          872

COMMENT   WHITE .VD HAS A CONSTANT VALUE OF 15
```

DISTANCE FACTORS

At this point the distance factors used to classify candidate target windows will be introduced. The correlation, L1, and L2 factors correspond to the normalized correlation, L1, and L2 distances linearly scaled into a 0-100 range, with 100 corresponding to a exact match and 0 corresponding to the distance to a constant valued scene. Consider the mappings

$$\begin{aligned} Nst &\rightarrow 0 && \text{for } Nst \le Nst|c \\ NL2 &\rightarrow 0 && \text{for } NL2 \ge NL2|c \\ NL1 &\rightarrow 0 && \text{for } NL1 \ge NL1|c \end{aligned} \tag{23}$$

and

$$\begin{aligned} Nst &= 1 \rightarrow 100 \\ NL2 &= 0 \rightarrow 100 \\ NL1 &= 0 \rightarrow 100 \end{aligned} \tag{24}$$

The functions to achieve these mappings are

$$\text{CFACTOR} = \begin{cases} 100[(\text{Nst} - \text{Nst}|\text{c})/(1 - \text{Nst}|\text{c})] & \text{Nst} > \text{Nst}|\text{c} \\ 0 & \text{else} \end{cases} \qquad (25)$$

$$\text{L2FACTOR} = \begin{cases} 100[1 - \text{NL2}/\text{NL2}|\text{c}] & \text{NL2} < \text{NL2}|\text{c} \\ 0 & \text{else} \end{cases} \qquad (26)$$

$$\text{L1FACTOR} = \begin{cases} 100[1 - \text{NL1}/\text{NL1}|\text{c}] & \text{NL1} < \text{NL1}|\text{c} \\ 0 & \text{else} \end{cases} \qquad (27)$$

A score is computed to take into account all three distance measures as follows:

$$\text{SCORE} = \sqrt[3]{(\text{CFACTOR} * \text{L2FACTOR} * \text{L1FACTOR})} \qquad (28)$$

Note that SCORE intentionally favors the Euclidean measure over the L1 metric (recall that the normalized correlation is an invertible function of the normalized Euclidean distance). The SCORE will always be a number from 0 to 100 inclusive. The behavior of the SCORE of a template measured against itself for various window shifts is given in Table II.

One of the problems in determining the distance functions is that they are computationally expensive, often infeasible, because of the large size arrays required for most applications. This is the case with many linear

TABLE II. BEHAVIOR OF DISTANCE FACTORS

TEMPLATE WINDOW [illegible] LENGTH= 45 ROWS TOP ROW= 90
(CENTERED ON [illegible]) WIDTH= 24 COLUMNS LEFTCOL= 97

SCENE FILE PIEMPHQ VD

CORRELATION PEAK (ROW, COLUMN)	SEARCH POINT (ROW, COLUMN)	TOP ROW	LEFT COLUMN	CORRELATE FACTOR	L2 FACTOR	L1 FACTOR	SCORE
1 [illegible]	[illegible]	89	96	47	29	45	40
	[illegible]	89	97	63	39	58	52
	[illegible]	89	98	37	22	42	33
	[illegible]	90	96	82	58	69	69
	[illegible]	90	97	100	100	100	100
	[illegible]	90	98	71	46	67	60
	[illegible]	91	96	65	41	50	51
	[illegible]	91	97	77	54	62	64
	[illegible]	91	98	54	32	47	43

processing algorithms. Indirect computational techniques based on unitary transforms permit more efficient lineaı processing than conventional methods. The next chapter introduces the Fourier transform as a linear processing tool.

III. LINEAR PROCESSING

An efficient method of linear processing is through the use of unitary transforms. A unitary transform meets the following three conditions [5: 232]:

1) It is a linear transformation;
2) Its operation is exactly invertible; and
3) Its operating kernel satisfies certain orthogonality conditions.

A unitary transform of particular importance in the field of image processing is the two-dimensional Fourier transform. In addition to its use as a linear processing tool, the Fourier transform provides a means of extracting features from images. For instance, the center or DC term is proportional to the average image brightness. The low-frequency terms contain the gross form information, while the high-frequency terms indicate the amplitude and orientation of the edges (the detail).

TWO-DIMENSIONAL DISCRETE FOURIER TRANSFORM

Consider a two-dimensional periodic sequence

$$\tilde{g}(x,y) = g(x+qM,y+rN) \qquad (29)$$

where q and r are integers, and M and N are the periods in the x and y direction. Such a sequence can be represented by a finite sum of exponentials in the form

$$\tilde{g}(x,y) = \frac{1}{MN} \sum_{fx=0}^{M-1} \sum_{fy=0}^{N-1} \tilde{G}(fx,fy)\exp[j2\pi(xfx/M+yfy/N)] \qquad (30)$$

where

$$\tilde{G}(fx,fy) = \sum_{x} \sum_{y} \tilde{g}(x,y) \exp[-j2\pi (xfx/M+yfy/N)] \qquad (31)$$

and $j=\sqrt{-1}$.

Note that $\tilde{G}(fx,fy)$ will have the same periodicity as the sequence $\tilde{g}(x,y)$. If a finite area sequence $g(x,y)$ is considered to be one period of $\tilde{g}(x,y)$, and $G(fx,fy)$ is taken to be one period of $\tilde{G}(fx,fy)$, then $g(x,y)$ and $G(fx,fy)$ will form a discrete Fourier transform pair. In equation form [7: 117],

$$g(x,y) = \begin{cases} \dfrac{1}{MN} \displaystyle\sum_{fx} \sum_{fy} G(fx,fy) \exp[j2\pi(xfx/M+yfy/N)] & \begin{array}{r} 0 \le x \le M-1 \\ 0 \le y \le N-1 \end{array} \\ 0 & \text{otherwise} \end{cases} \qquad (32)$$

$$G(fx,fy) = \begin{cases} \displaystyle\sum \sum g(x,y) \exp[-j2\pi(xfx/M+yfy/N)] & \begin{array}{r} 0 \le fx \le M-1 \\ 0 \le fy \le N-1 \end{array} \\ 0 & \text{otherwise} \end{cases} \qquad (33)$$

The notation to be used to indicate a Fourier Transform pair is

$$g(x,y) \longleftrightarrow G(fx,fy) \qquad (34)$$

Equivalently,

$$F\{g(x,y)\} = G(fx,fy) \qquad (35)$$

and

$$F^{-1}\{G(fx,fy)\} = g(x,y) \qquad (36)$$

Having defined the Fourier transform, two theorems for use in later developments will be stated without proof [7: 110].

SHIFT THEOREM

$$\text{If } t(x,y) \longleftrightarrow T(fx,fy), \tag{37}$$

$$\text{then } t(x-m\ ,y-n\ ) \longleftrightarrow \exp[-j2\pi(mfx/M+nfy/N)]T(fx,fy)$$

REVERSAL THEOREM

$$\text{If } t(x,y) \longleftrightarrow T(fx,fy),$$

$$\text{then } t(-x,-y) \longleftrightarrow T^*(fx,fy),$$

$$\text{where } T^* \text{ is the complex conjugate of } T. \tag{38}$$

The convolution theorem suggests a method for performing correlation in the Fourier domain. It will now be stated with its proof, modeled after a proof for continuous signals [4: 307].

CONVOLUTION THEOREM

$$\text{For } s(x,y) \longleftrightarrow S(fx,fy) \text{ and } t(x,y) \longleftrightarrow T(fx,fy),$$

$$F\{s(x,y)*t(x,y)\} = S(fx,fy)T(fx,fy) \tag{39}$$

Proof:

By definition,

$$F\{\sum_x\sum_y s(x,y)t(m-x,n-y)\} = \tag{40}$$

$$\sum_m\sum_n[\sum_x\sum_y s(x,y)t(m-x,n-y)]\exp[-j2\pi(mfx/M+nfy/N)]$$

Interchanging the summation order,

$$= \sum_x\sum_y s(x,y)\{\sum_m\sum_n t(m-x,n-y)\exp[-j2\pi(mfx/M+nfy/N)]\}$$

Application of the shift theorem gives

$$= \sum_{x} \sum_{y} s(x,y)\{\exp[-j2\pi\,(xfx/M+yfy/N)]T(fx,fy)\} \qquad (42)$$

$$= \quad S(fx,fy)T(fx,fy) \qquad (43)$$

By the reversal theorem, it follows that

$$F\{\sum_{x} \sum_{y} s(x,y)t(x-m,y-n)\} = S(fx,fy)T^*(fx,fy) \qquad (44)$$

Thus, an indirect method for performing cross-correlation is given by

$$Rst(m,n) = F^{-1}\{S(fx,fy)T^*(fx,fy)\} \qquad (45)$$

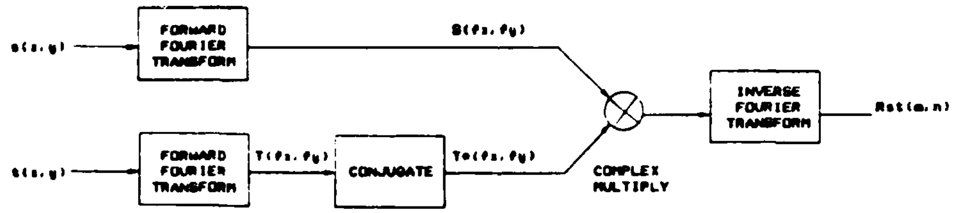

s(x,y)
FORWARD FOURIER TRANSFORM
S(fx,fy)
t(x,y)
FORWARD FOURIER TRANSFORM
T(fx,fy)
CONJUGATE
T*(fx,fy)
COMPLEX MULTIPLY
INVERSE FOURIER TRANSFORM
Rst(m,n)

FIGURE 2: INDIRECT CORRELATION METHOD.

Highly efficient algorithms exist for computing the discrete Fourier transform of a finite-duration sequence. Thus, a computationally reasonable implementation of the correlation of two sequences is given by the use of the discrete Fourier transform. Note that in many applications, the same template is correlated with many different input scenes, so that $T^*(fx,fy)$ needs to be computed just once and stored.

ARRAY EXTENSION FOR LINEAR CORRELATION

Care must be taken in choosing sequence and transform lengths. Consider the linear correlation between two N-point sequences $R(m) = \sum s(x)t(x-m)$, where R(m) will have up to 2N-1 non-zero points. The indirect correlation method using N-point discrete Fourier transforms will result in an N-point sequence, which is the circular correlation of the input sequences. To obtain the linear correlation, the discrete Fourier transforms must be computed on the basis of 2N-1 or more points, with the input sequences extended with at least N-1 zeros. In general, for s(x) of length S1 and t(x) of length T1, the indirect linear correlation method requires that discrete Fourier transform be computed on the basis of at least S1+T1-1 points.

For the two-dimensional case, the sequence arrays are extended as follows (See Figure 3) [5: 288]:

1)Imbed the T1xT2 template image sequence in the lower right quadrant of an all zero M1xM2 matrix;

2)Imbed the S1xS2 input scene in the upper left quadrant of an all zero M1xM2 matrix;

3)Compute all discrete Fourier transforms on the basis of M1>S1+T1-1 and M2>S2+T2-1 to avoid wrap-around error.

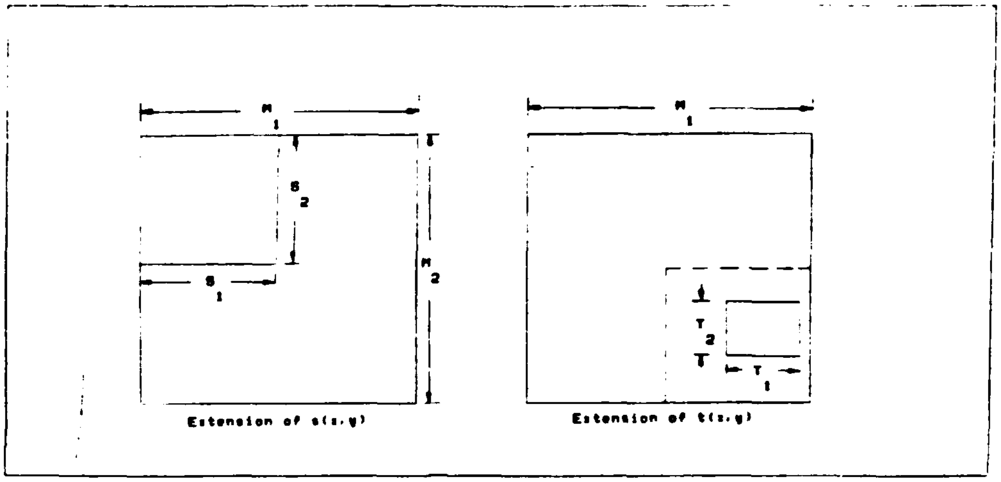


FIGURE 3 ARRAY EXTENSION FOR LINEAR CORRELATION

By taking advantage of the speed gained by implementing the indirect correlation method, the ability to normalize the scene search window during the correlation process is lost. This inability is a major shortcoming of the indirect correlation method. An alternate normalization scheme must be implemented to approximate the window-by-window normalization method.

IV. NORMALIZATION SCHEME

Consider dividing the scene to be normalized into a grid of rectangles. A compromise between global normalization and search window normalization would be to divide each of the scene values within a given rectangle by some constant that is proportional to the square root of the energy of that rectangle. This normalization scheme may be implemented as in Figure 4.

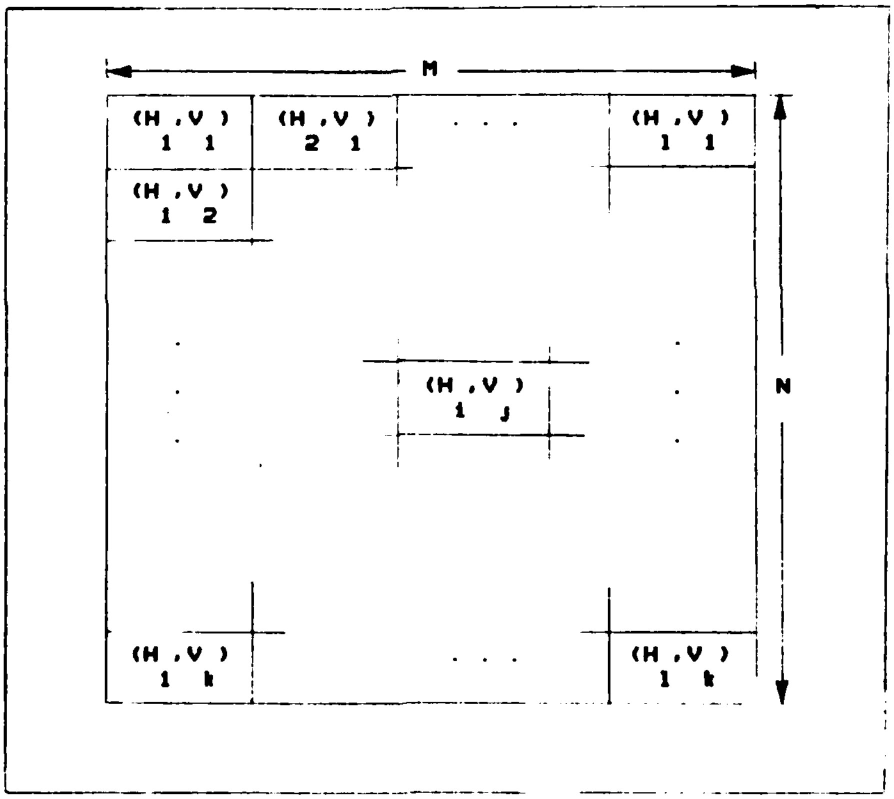


FIGURE 4: NORMALIZATION GRID SETUP

(Hi,Vj) -- grid rectangle in grid row j, grid column i

N -- of pixel rows

M -- of pixel columns

(i,j,M,N,N/k, and M/l are all integers).

Define

$$Eij \triangleq \sum_x \sum_y s^2(x,y) \qquad (46)$$

for $(i-1)M/l + 1 \leq x \leq iM/l$

and $(j-1)N/k + 1 \leq y \leq jN/k$

Then for

$s'(x,y)$ -- normalized array value

Nij -- normalization coefficient

Eij -- energy in grid rectangle (Hi,Vj)

$$s'(x,y) \triangleq s(x,y)/Nij \qquad (47)$$

where

$$Nij = a\sqrt{Eij} \qquad (48)$$

Some thought must be given to the size of the rectangles chosen, for as the size is increased, the clutter energy-to-target energy ratio is also increased (ideally this ratio should be zero). As the size of the rectangles is decreased, the scene begins to lose contrast as the normalized values asymptotically approach a constant value (namely 1 when the rectangle size is 1x1). Another problem accompanying the decrease in rectangle size is the possibility of sectoring part of a target into separate rectangles increases; this could have a deleterious effect on the cross-correlation function. See Figure 5 for an illustration of a normalized scene.

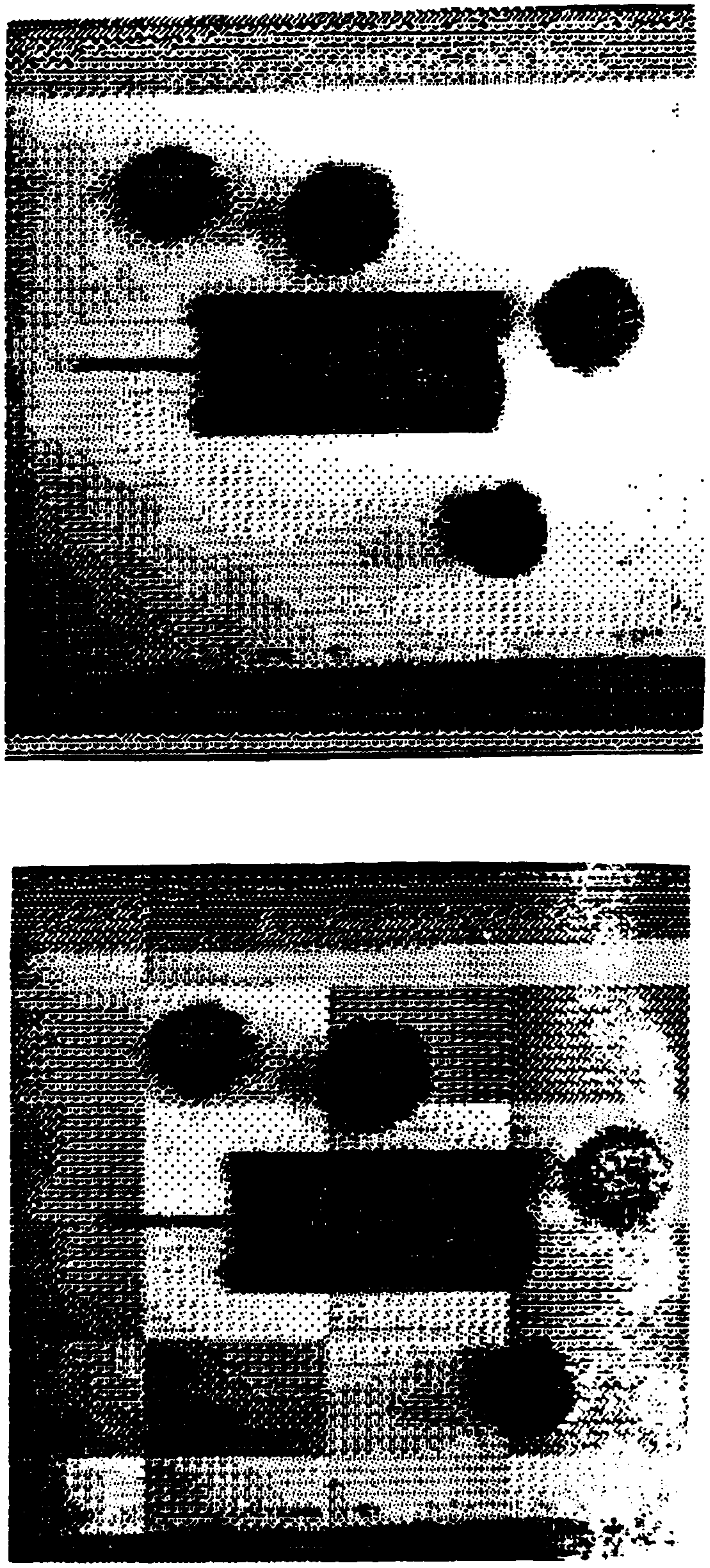


FIGURE 5: TOP: SCENE PTANKD2.
BOTTOM: SCENE NORMALIZED WITH A 4X6 GRID.

Consider the following two cases, shown in Figure 6, in which a 3x6 normalization grid has been chosen.

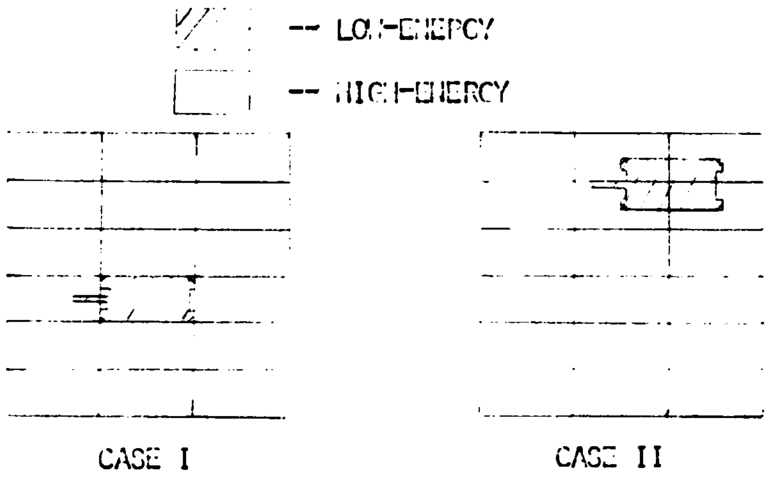


FIGURE 6: EXAMPLE OF TARGET SECTORING PROBLEM.

In case I, the target will be satisfactorily normalized against the high energy background, allowing a successful global search by correlation. In case II, the clutter-to-target area is large for all grid rectangles. The high energy areas dominate, yielding false peaks in the correlation function.

In the next chapter, the software necessary to process the images and test the target detetection process is described. Also, the detection process is further discussed.

V. SOFTWARE DESCRIPTION

The software that was used in this study is described in this chapter. The programs (indicated by capitalized titles) have been grouped into the following categories:

1)Image input and output;

2)Scene and template synthesis;

3)Correlation implementation;

4)Process evaluation; and

5)Support subroutines.

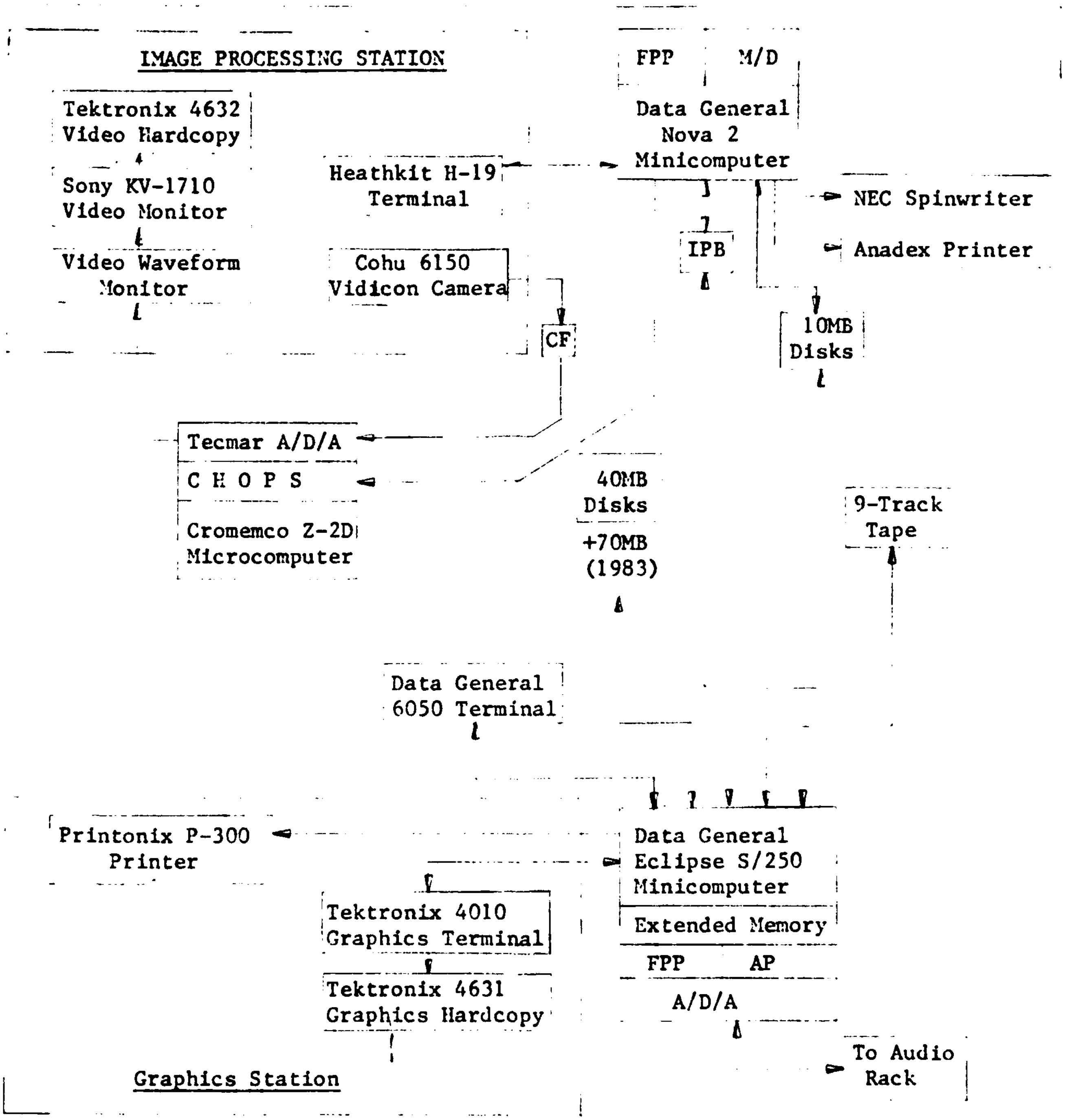
All source code that was generated during this thesis effort and added to the AFIT Signal Processing Laboratory software archives is included in the appendix. All programs are written in Data General (DG) Fortran 5 (except for VIDEO7 and NMOVE, which are written in Fortran IV). All programs were written by James Cromer, with the exception of PLTTRNS, INVERSE, and DIRECT (by Ronald Schafer), and IOF, UNPACK, and REPACK (by Robin Simmons).

IMAGE I/O

Before a digital computer can be used to analyze an image, the image must first be converted to a form usable by the computer. Specifically, the image must be represented in some sense by an array of numbers. The process used to obtain this array is known as digitization, in which some image parameter is sampled and quantized at points throughout the scene. In this study, the achromatic brightness, or gray level, is sampled in a 256 column by 256 row quadruled grid format, and quantized into one of 16 levels (4-bit digitization). The resulting array values are referred to as "pixels", short for "picture elements."

The equipment used in the process include a standard video monitor, a Cohu 6150 vidicon camera with 6950 camera controls, and 3 Tecmar digitizer boards (A/D converter, direct memory access, D/A). The Tecmar digitizer is interfaced with a DG NOVA 2 processor via a CROMEMCO Z-80 based microcomputer. The NOVA terminal can be used to communicate with the A/D/A converter with the Fortran callable subroutine CHANNEL, developed in an earlier AFIT thesis [8]. High speed processing can be performed on the digitized images with the powerful DG ECLIPSE S/250 minicomputer. Both DG machines are 16-bit processors. See Figure 7 for a schematic of the equipment layout.

The program which controls image input (digitization) and output (display) through CHANNEL is VIDEO7. When running VIDEO7 and the input option is chosen, seven digitized versions of an input image are stored in files named "A0" through "A6", which can be averaged together later. When the output option is chosen, the user is given the choice to display from one to ten files named "A0" through "A(n-1)", where n is the number of files to be displayed. In addition to the main purpose of checking images before averaging, n



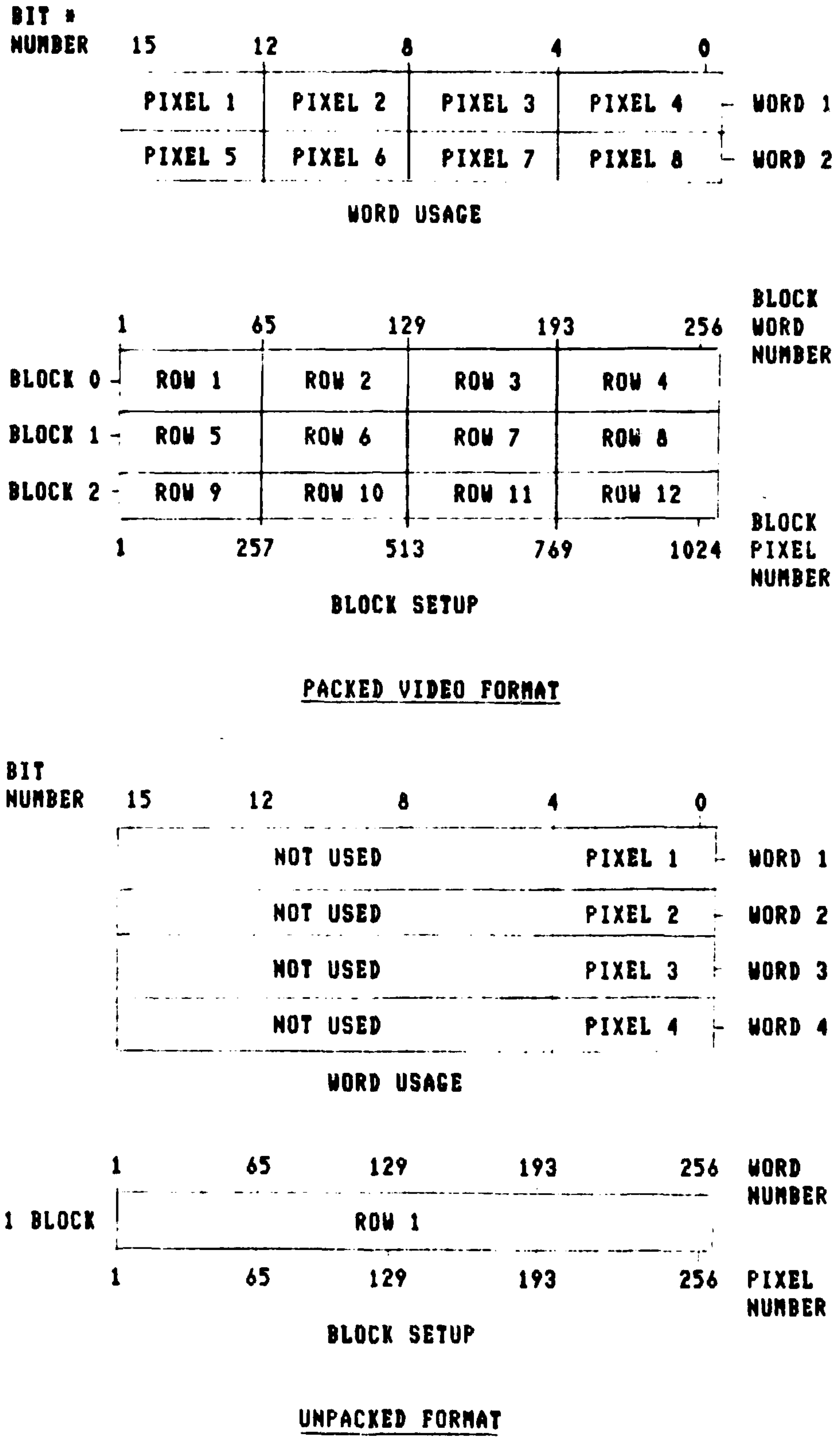
CF	--	Low Pass Filter
M/D	--	High-speed Multiply and Divide unit
FPP	--	High-speed Floating Point Processor
AP	--	High-speed Array Processor
IPB	--	Inter-Processor Buffer
A/D/A	--	Analog-to Digital and Digital-to-Analog Converters

Figure 7: Equipment Layout of the AFIT Signal Processing Laboratory

images of interest may be easily presented in sequence for demonstrations by re-naming the images "A0" through "A(n-1)." The third mode of VIDEO7 allows the user to display an existing file any number of times consecutively. This mode is used when the D/A converter malfunctions, and usually several attempts to view an image must be made before a satisfactory image is displayed.

Files created by VIDEO7 are written to disk in what is referred to as "packed video form." The file is "packed" because 4 pixels are stored in each 16-bit word. Packed form is ideal for minimizing storage requirements, while posing only minor processing inconveniences. In packed form, video files will be 64 blocks long, where one block is 256 16-bit words. Thus one image requires only 32K bytes of memory. Note that each block holds 4 packed video rows. As a result, the processing programs operate on multiples of 4 rows at a time between RDBLK calls. The fastest way to transfer data from disk storage to core memory is by the RDBLK call, which passes data in the data channel mode. The data channel mode of moving data does not require program control once a transfer is initiated. Figure 8 shows more clearly the relation between packed and unpacked forms.

The seven digitized images created by VIDEO7 can be averaged to produce an output image that has an improved signal to noise ratio. The program that does this averaging



*Bit position numbering convention used by ISET and ITEST.

FIGURE 8: PACKED AND UNPACKED VIDEO FORMATS.

is called QUICKAVE7, which creates an output file named "AVERAGE7.VD", where the .VD extension indicates a video file. The seven files to be averaged are assumed to be named "A0" through "A6."

It is often desirable to produce a hard copy of a digitized image. One of the ways this can be done is by displaying an image through VIDEO7, then activating the Tektronix 4632 Video Hard Copy Unit, which makes a photocopy of the image being sent to the video monitor. This method is acceptable most of the time, but for numerous reasons it is necessary to produce hard copies of stored images with the Printronix P-300 lineprinter. The program originally written to do this is DISPLAY, by Robin Simmons in an earlier thesis [9]. DISPLAY used 3x3 dot patterns to simulate the 16 gray-levels, which resulted in two shortcomings. Distortion occurs in the picture because the P-300 horizontal dot density is less than the vertical density, resulting in a 1.2:1 aspect ratio. Also the 3x3 patterns do not fully take advantage of the 16 gray-levels available. DISPLAY was modified to solve these problems by using a combination of 3x3 and 3x4 dot patterns. Up to four different dot patterns are used per gray-level, instead of just one, and the aspect ratio is very nearly 1:1. Other modifications include allowing the user to choose the number of rows to be displayed along with the starting row. The run time for an 11x13 image hard copy was reduced to less than 90 seconds, down from 5-6 minutes. The modified program is called PICTURE.

SCENE and TEMPLATE SYNTHESIS

After an image is digitized and stored, the next step is to create a template or scene to be used in the correlation process. To do this programs were written to improve the image, create a scene or template by combining images, and to put the images into the correct format for the correlation process.

The program REMOVE was written to perform a 3x3 pixel mask processing of an image for the purpose of noise removal. The main program handles the bookkeeping of passing the three rows to be operated on to the subroutine TEST3, which produces the noise-removed output row. The subroutine UNPACK2 is used to unpack the video rows from four pixels per word to one pixel per word. REMOVE was not used extensively, but is included to demonstrate an efficient method to perform mask processing. The mask function can be changed by modifying TEST3. TEST3 presently computes the difference between the center pixel value and the average value of the surrounding pixels. If the difference is greater than some threshold, the center pixel value is modified accordingly.

If REMOVE and QUIKAVE7 fail to produce a satisfactory image, a histogram can be generated, and then modified to enhance the image. The program to produce the histogram is called EVIDHIST (the "E" indicates an Eclipse only program). TONER modifies the histogram by a mapping function of the type: 0 -> a, 1 -> b, . . . , 15 -> p, where the user defines the new values of "a" through "p". TONER is used to increase the contrast or raise the average brightness level when deficiencies occur due to A/D or camera gain misadjustment.

To create a template from a scene with a target in it, a "window" is placed over the target information, and the background is set to some constant value (usually 0 or 15).

Program NMOVE allows the user to specify a template scene file, a background file, and a combined filename. The template window size and position are variable, as is the combined window position. Figure 9 demonstrates the capability of NMOVE.

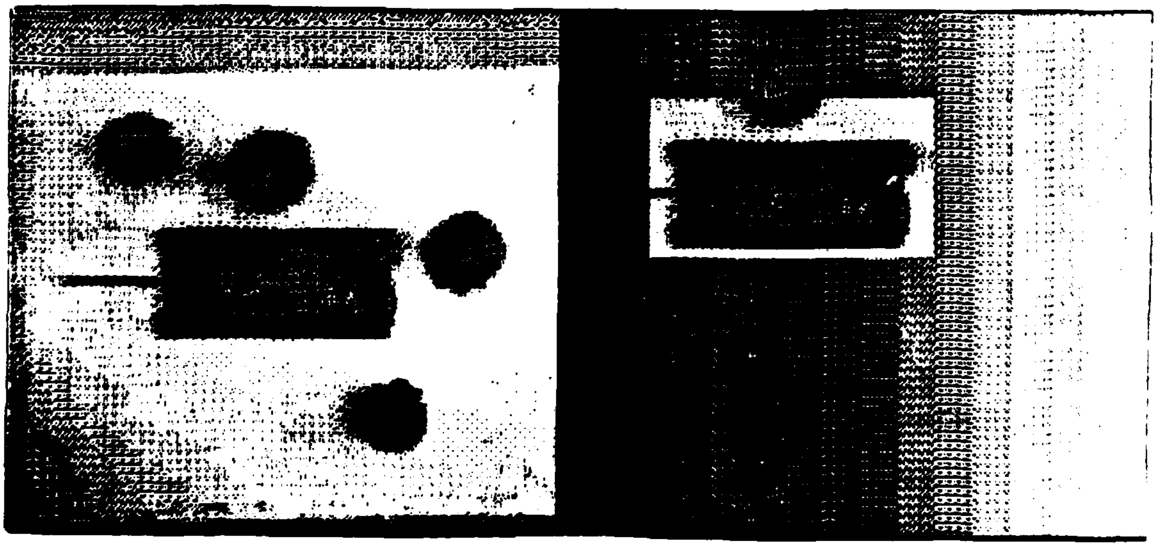


FIGURE 9: LEFT: NMOVE INPUT SCENE. RIGHT: SCENE CREATED BY NMOVE. THE BACKGROUND SCENE IS THE 16 GRAY-LEVEL BARPATTERN.

The next step requires that the "negative" image be formed, using the equation NEGATIVE = 15 - POSITIVE. This forces the expected high energy background to become a low energy background, improving the correlation results. TONER can also be used for this purpose, but NEGATE is used in macro files, as it requires no user input.

The last step before the correlation sequence is to do a four-to-one reduction to imbed the 256x256 scene into the upper left quadrant (lower right for template) so that linear correlation will be obtained. Program REDUCE does this by averaging four pixels to create one output value. This reduction is also effectively a low-pass filtering operation. See Figures 10 and 11 for flowcharts of the scene and template synthesis processes.

CORRELATION IMPLEMENTATION

The correlation method used is to perform a multiplication in the frequency domain, then inverse transform the product, resulting in a correlation in the spatial domain. The complex arrays multiplied bave been obtained from Fourier transforming normalized image arrays.

The normalization scheme used is the rectangle grid normalization described in Chapter 4. The program which carries out the normalization is suitably named NORMALIZE. The number of grid rows and grid columns are chosen by the user. Only the upper left or lower right quadrants are

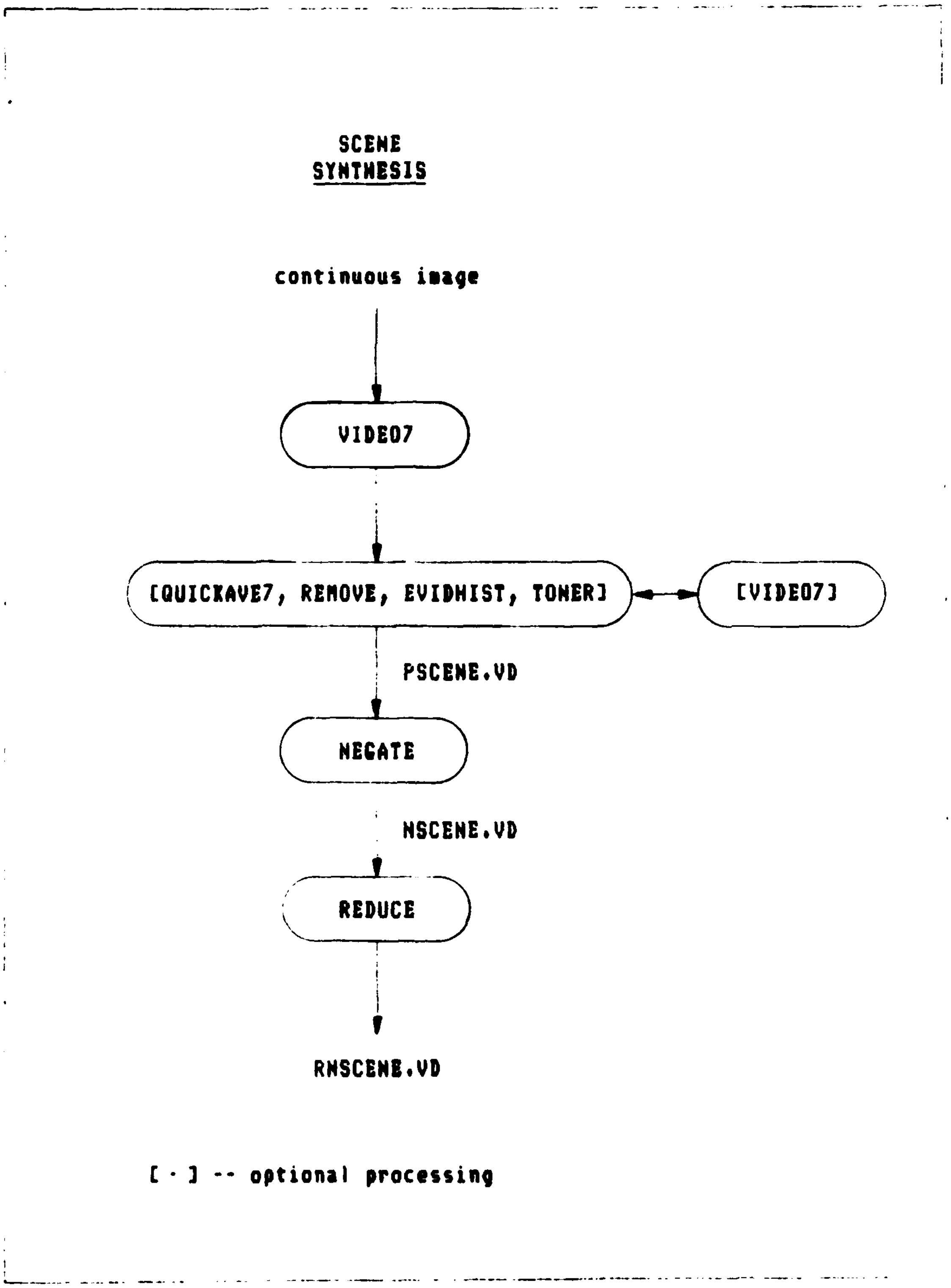


FIGURE 10: FLOWCHART OF SCENE SYNTHESIS.

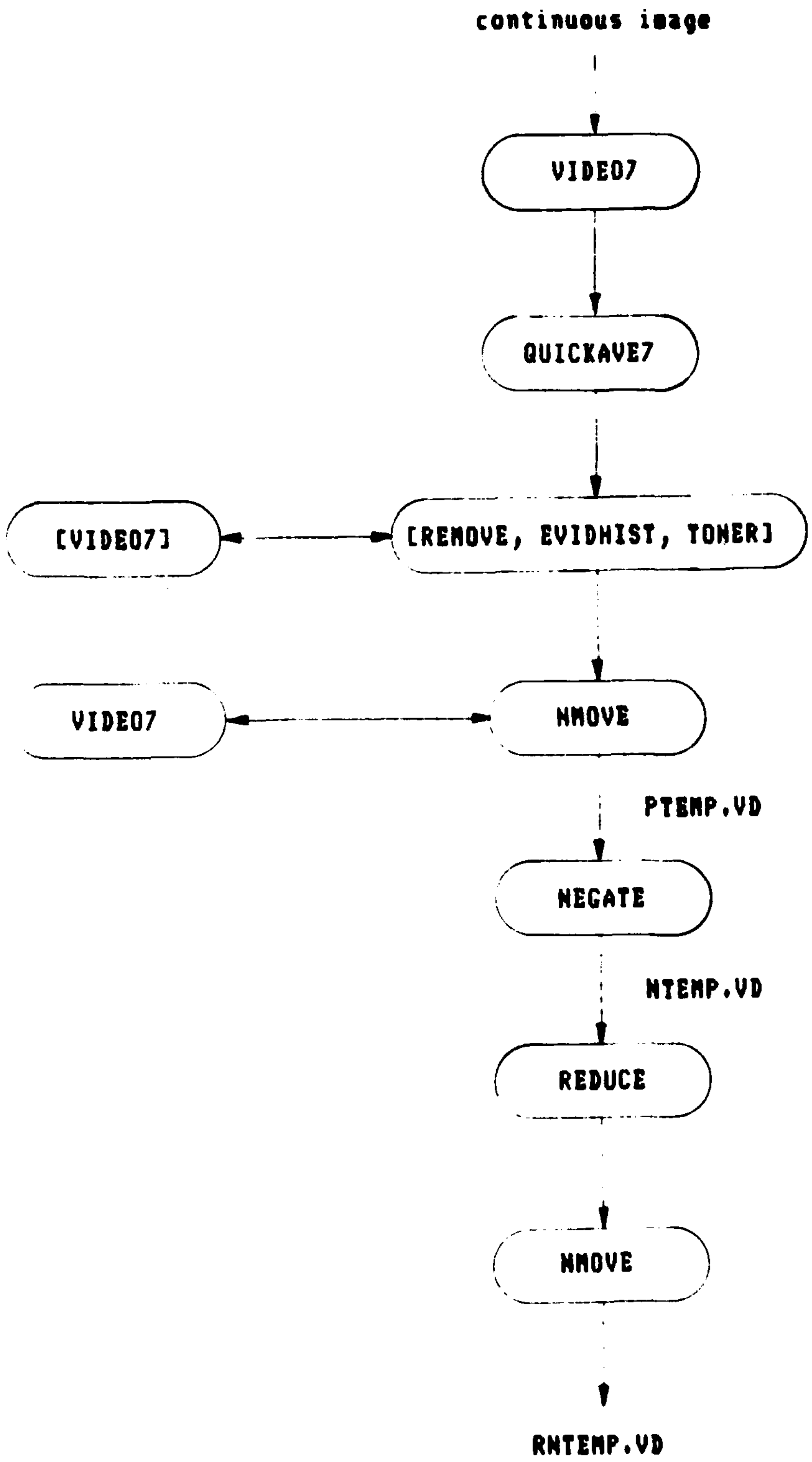


FIGURE 11: FLOWCHART OF TEMPLATE SYNTHESIS.

normalized; the remainder of the array is set to zero. The input file is a 256x256 packed video, and the output is a 256x256 complex file.

After the normalized scene and template complex files have been created, they are Fourier transformed by the program DIRECT. The resultant arrays are then complex multiplied by CMULTIPLY, and the product is inverse transformed by INVERSE. INVERSE and DIRECT are Eklundh FFT algorithm based programs written by Ron Schafer.

The file created by INVERSE is a complex array with the imaginary part zero (since the scene and template arrays are always real), and with the real part having values between zero and two. CTOI conserves file space by converting the complex array to integer form by multiplying each real number by 16384 to take advantage of the 16 bit word. The integer file uses only one-fourth as much disk memory as a complex file.

The program CTOV can be used to convert a complex file (imaginary part assumed zero) into a video file. It performs a linear scaling of the input file into a 0-15 range. CTOV was used (along with PICTURE) to display the complex normalized scene of Figure 5.

The last step in the correlation process is to combine the results of several correlations between a scene and a set of templates. Program IMULTIPLY computes the geometric mean of two correlation functions by determining the square root of the product of the input arrays. An example of when IMULTIPLY may be used is when the correlation functions created from the left-half and the right-half of a template are to be combined. IMULTIPLY can also be used to combine the positive correlation (positive scene with positive template) with the negative correlation (negative scene with negative template). See Figure 12 for a flowchart of the correlation implementation.

PROCESS EVALUATION

The function resulting from the correlation process next needs to be evaluated. The correlation function can be evaluated by viewing it, or by a numerical analysis of its peaks.

PLTTRNS, by Schafer, enables the user to view 3-D, contour, and row plots on the Tektronix 4010 graphics terminal. The capabilities of PLTTRNS are enhanced by ITOC, which, among other tasks, converts integer files to complex files usable by PLTTRNS. See the source code listing in the appendix for further information on the use of ITOC.

The plots generated by ITOC and PLTTRNS give a rough idea of the success of the global search. PEAK gives

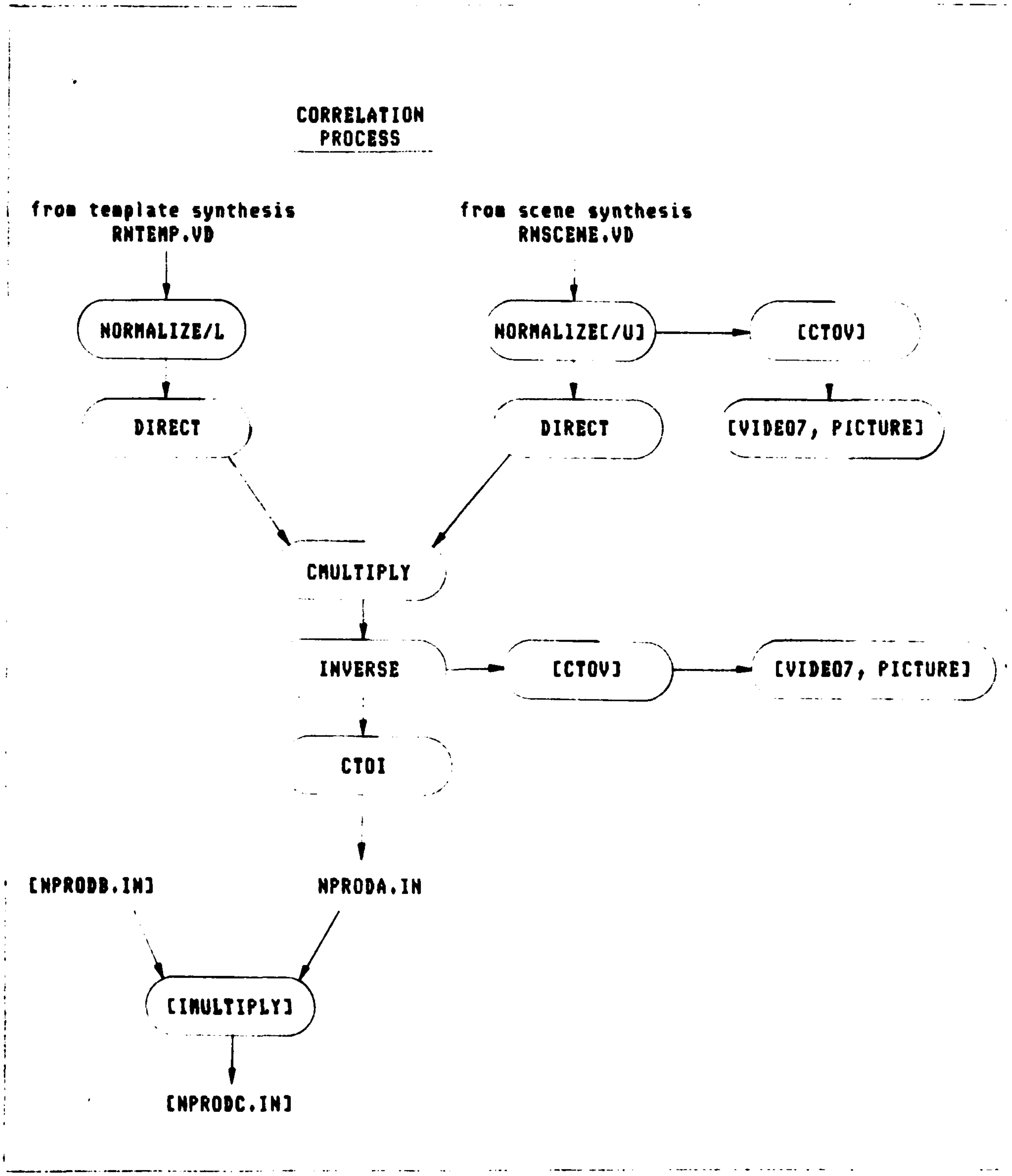


FIGURE 12: FLOWCHART OF CORRELATION IMPLEMENTATION.

quantitative information on up to ten supra-threshold correlation plane peaks: location, width, length, and value. More peak information is generated internally by PEAK if further peak discrimination is desired. As the correlation functions generally are not monotonically non-decreasing functions toward the absolute peak value, the selection of a threshold value is not a straight-forward task.

DISTANCE uses the peak locations found by PEAK to calculate a set of distance factors between the template and the scene window. The distance factors are based on the L1 and L2 norms, and take into account the scene and template energies, the template average pixel value, and the window size. The factors can be computed for the 256x256 pixel original images, or the 128x128 pixel reduced images used in the correlation process. The factors are scaled into a 0-100 range. See Figure 13 for a flowchart of the evaluation process.

SUPPORT SUBROUTINES

Several subroutines are common to many programs; they will be briefly described below.

TIMER is used to measure the execution time of a program in hours, minutes, and seconds. The first call to TIMER starts the "stopwatch," and the second call stops it. The run time is printed on the console.

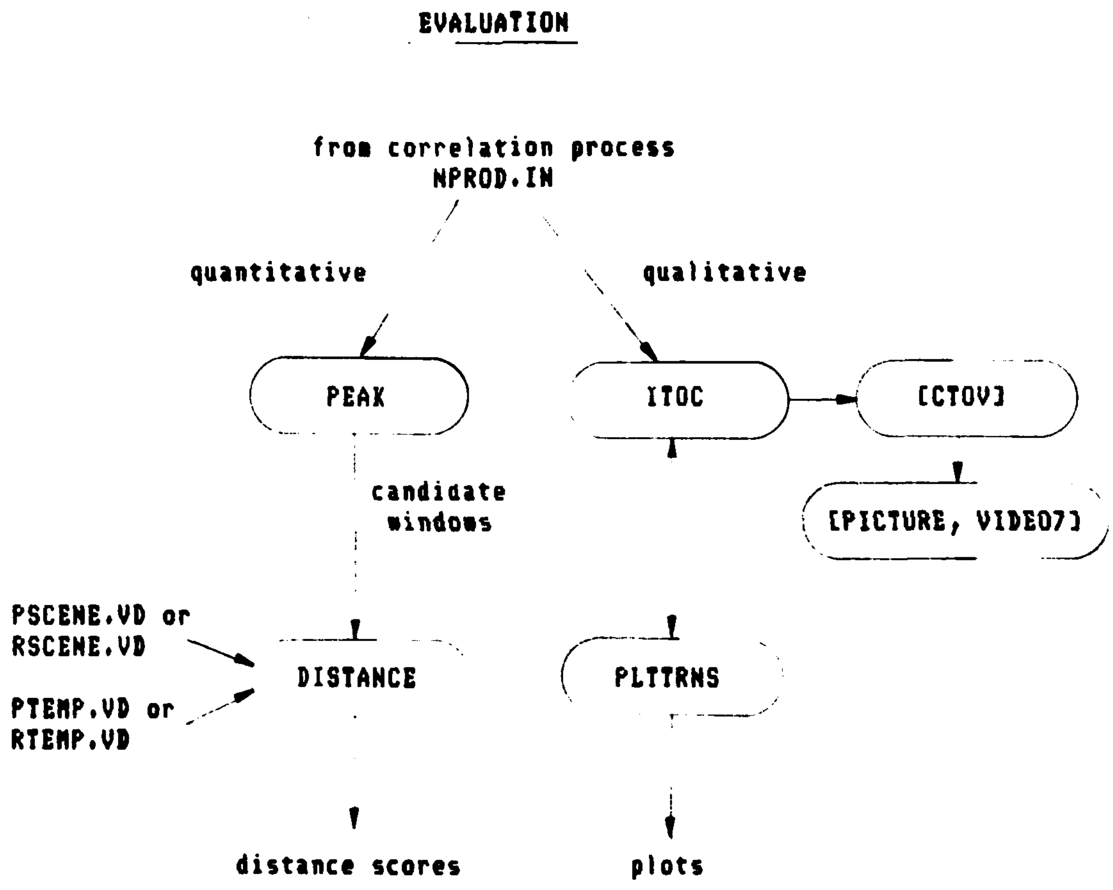


FIGURE 13. FLOWCHART OF EVALUATION PROCESS.

IOF is used to allow switches and filenames to be read in the execution command line. IOF is a powerful device when macro (or automatic) programs are run.

REPACK is the routine to convert unpacked arrays into packed arrays before they are written to disk. UNPACK converts packed arrays read from disk into unpacked to be processed.

XRDBLK is used to read a packed block (256 words) from disk, unpack it, and return the unpacked array (1024 words) to the calling program.

In the next chapter, observations will be made on some results of the correlation process.

VI. OBSERVATIONS

In this chapter, the behavior of the detection process is briefly discussed, and demonstrated with several appropriate examples. In particular, the problems of dealing with the clutter energy, variable target illumination, window positioning errors, and window classification are considered. (See the appendix for more test results.)

CLUTTER ENERGY

The results of some early testing led to several modifications of the correlation process. Consider the cross-correlation functions (CCF) between a template and a scene with a target in it, shown in Figure 14, and the CCF between a template and a scene without a target, shown in Figure 15. The template and both scenes have energies of unity (globally normalized). In neither case are there clearly defined peaks in the CCF to suggest possible target locations. The maximum function values in both cases corresponded to windows of light background only, rather than to "interesting" objects such as trees or tanks. This behavior is attributed to the dominance of the high-energy background. Relatively low energy objects will not correlate well with the template, regardless of their forms. This failure is a classic weakness of the non-normalized CCF.

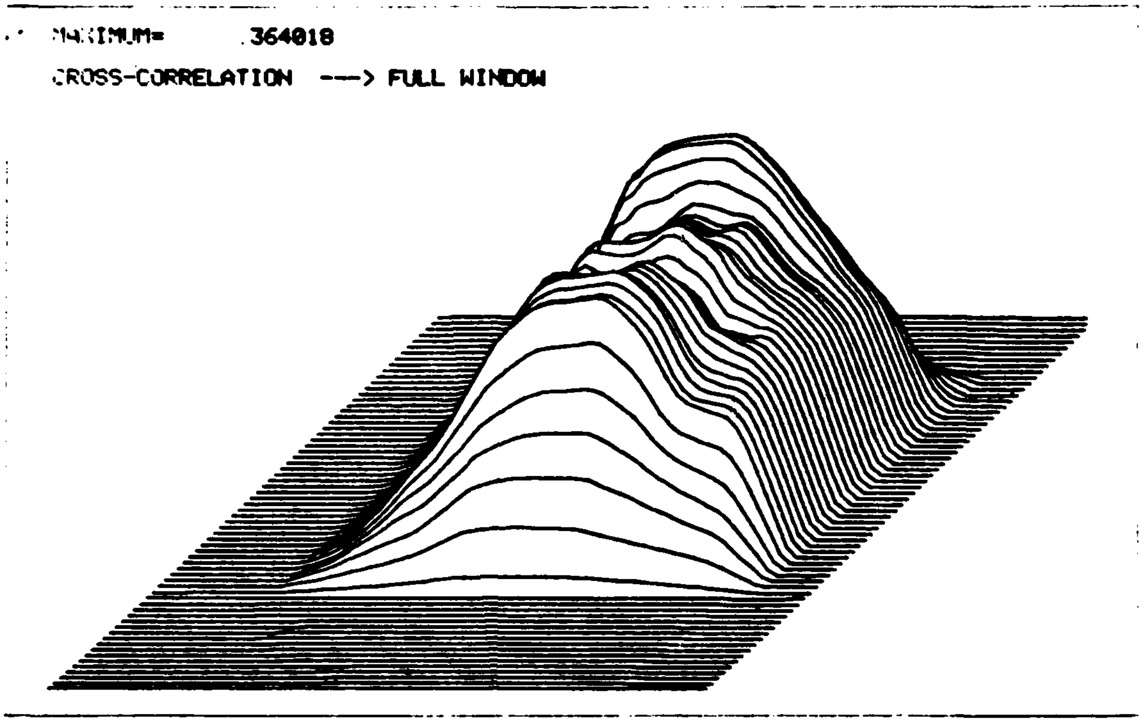


FIGURE 14: POSITIVE CCF FOR TANK SCENE TEST.VD AND TEMPLATE H3. GLOBAL NORMALIZATION USED.

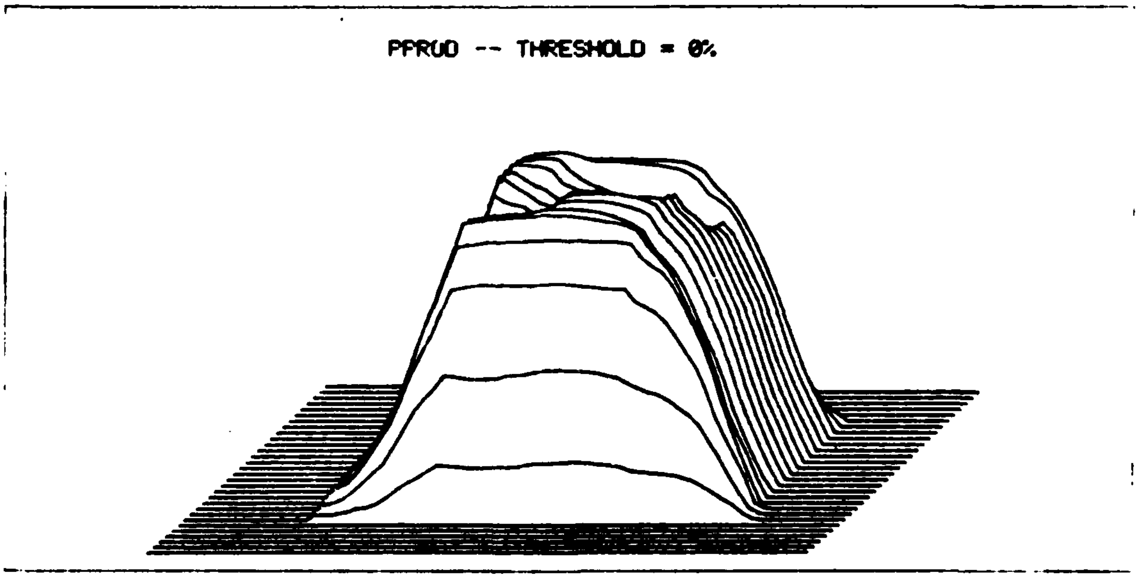


FIGURE 15: POSITIVE CCF FOR SCENE I3 AND TEMPLATE H3.

To take advantage of the a priori knowledge that the object of interest will have low-energy content, the CCF's were computed using the negatives of the scene and template. These are shown in Figures 16 and 17. The peaks of the functions are much more distinct than those of the positive CCF's. Notice especially the center peak of Figure 16, which corresponds to a tank in the scene.

The negative CCF's appeared to be an improvement, but two problems still remained. First, the digitizer noise of columns 1-8 resulted in the line of peaks seen at the right hand side of the negative CCF's. This line can clearly be seen in the top 20% contour plot in Figure 18. The second problem was that the false peaks in Figure 16 had values as large as the true peak (the true peak is the single peak corresponding to the target). One method to overcome these problems is to investigate each candidate peak by recomputing the CCF for a smaller sized scene window corresponding to that peak. Each new CCF is then investigated for the occurrence of distinct peaks, and the process continues until one window in the scene is chosen as being most likely to contain a target. Then a decision is made as to the content of the window.

The process of iteratively "windowing-in" on a target can be successful because the target-to-clutter area ratio

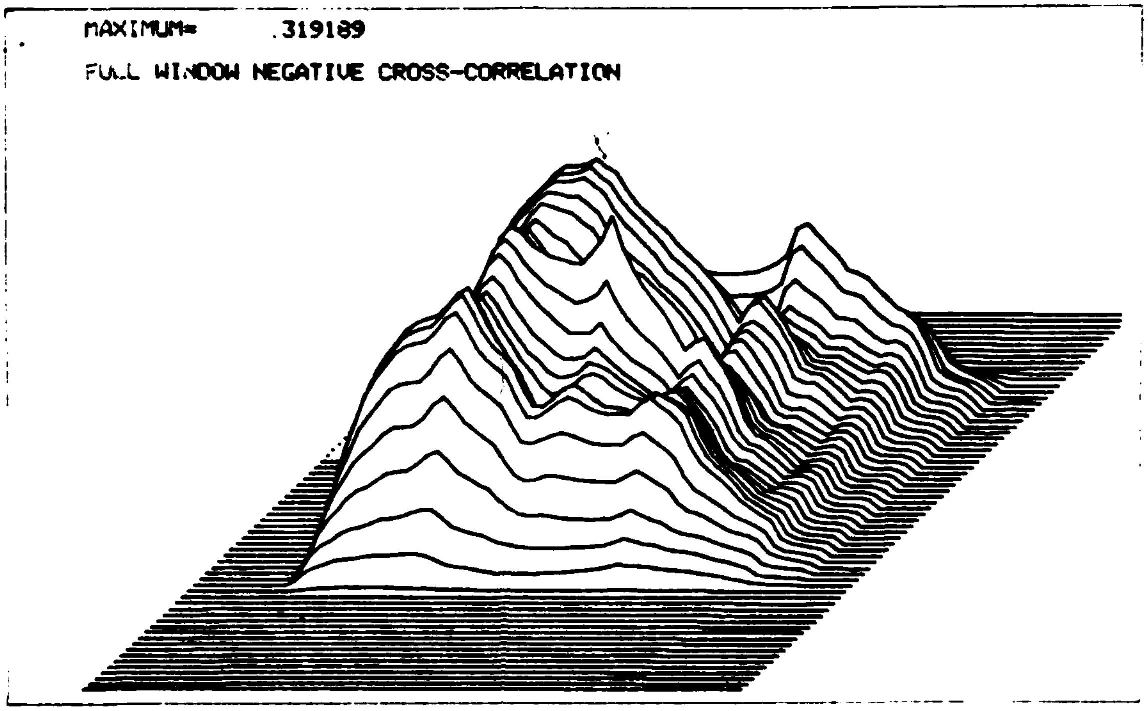


FIGURE 16: NEGATIVE CCF FOR TANK SCENE TEST.VD AND TEMPLATE H3. TARGET-TO-CLUTTER AREA RATIO IS 0.12.

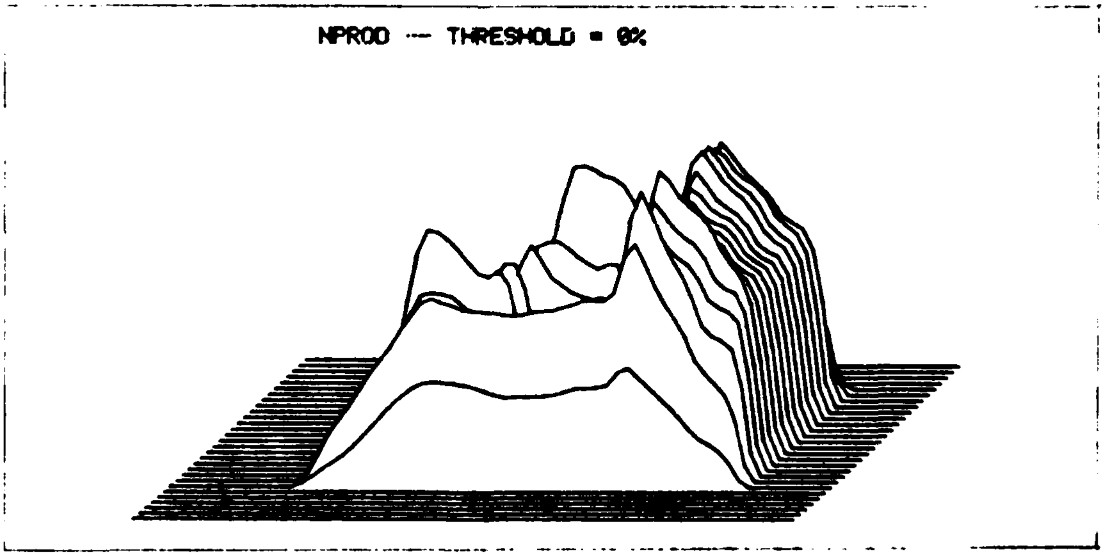


FIGURE 17: NEGATIVE CCF FOR SCENE I3 AND TEMPLATE H3.

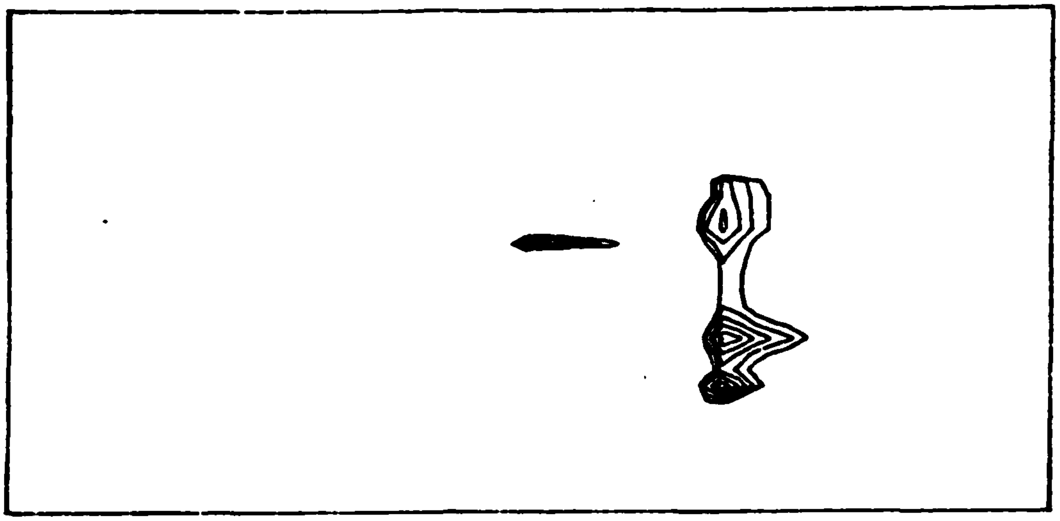


FIGURE 18: TOP 20% CONTOUR PLOT OF FIGURE 17.

increases with each successive CCF computed. Consider Figure 16, in which the target-to-clutter area ratio was 0.12. Notice (in Figure 19) the dramatic improvement obtained by increasing the target-to-clutter area ratio to 0.40.

An alternate method of choosing candidate windows is desired, as the windowing process can become computationally prohibitive for large scene areas. For this reason, the scene was locally normalized by grid blocks, in hopes of obtaining a distinct true peak with just one CCF calculation. In all test cases a 4x6 normalization grid was used. Rows 121-128 and columns 1-8 of the reduced scenes were set to zero prior to normalization to remove the possibility of digitizer noise dominating the correlation function. The effect of these two changes can be seen by comparing Figure 20 with Figure 17.

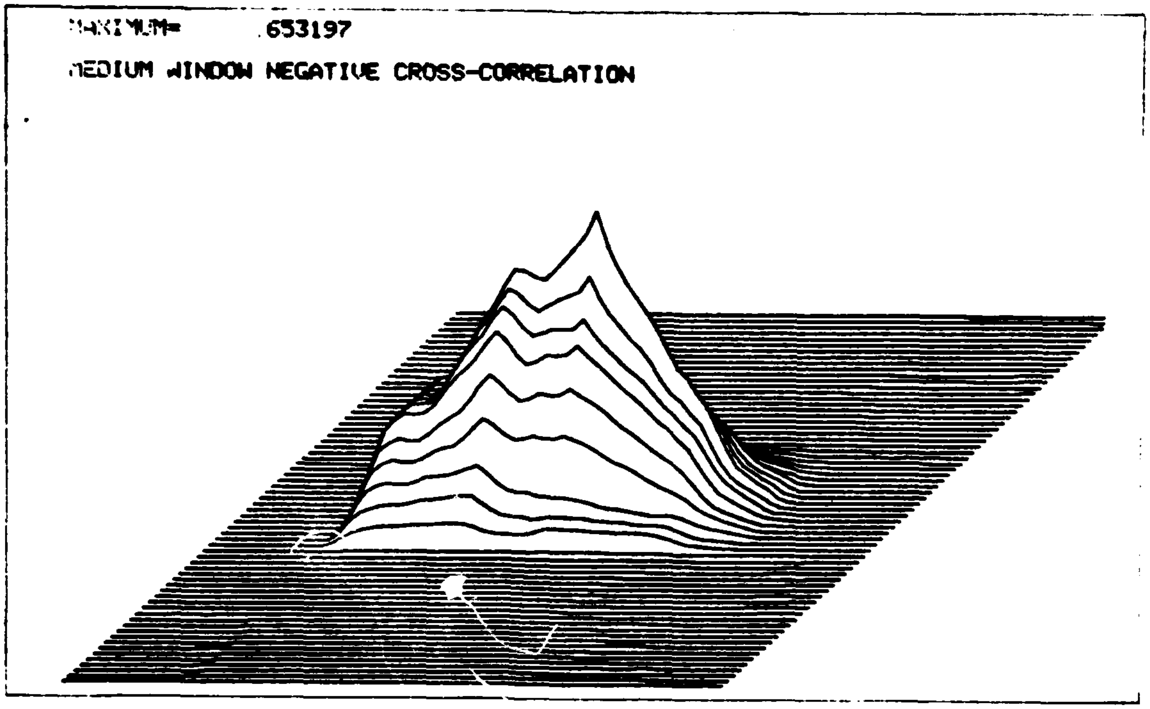


FIGURE 19: NEGATIVE CCF FOR SCENE TEST.VD AND TEMPLATE H3, WITH TARGET-TO-CLUTTER AREA RATIO OF 0.40.

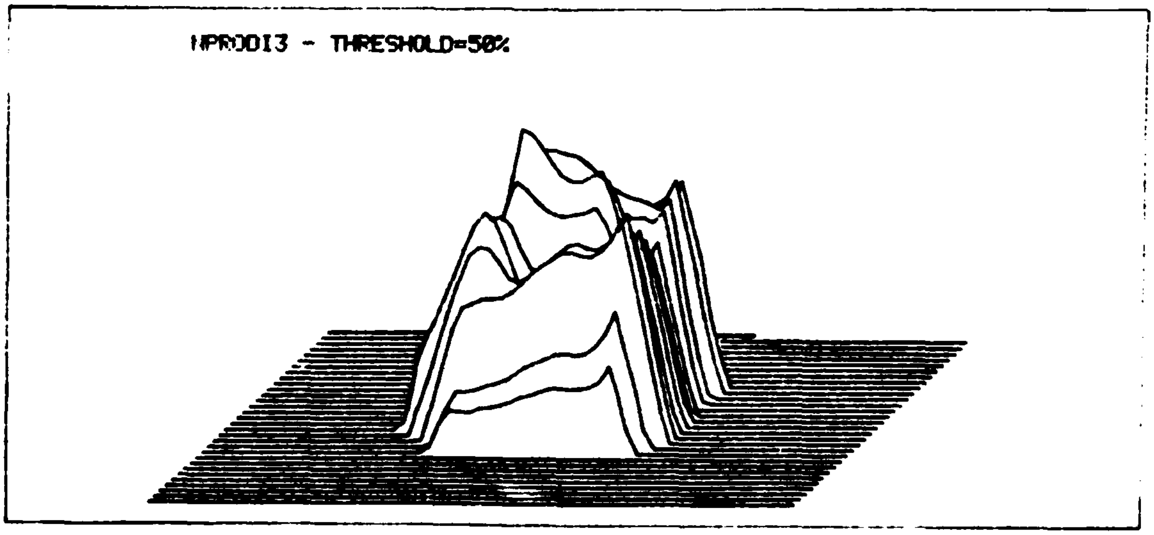


FIGURE 20: TOP 50% OF NEGATIVE CCF FOR SCENE I3 AND TEMPLATE H3. GRID NORMALIZATION USED.

The distance factors corresponding to the peaks of the CCF of Figure 20 are given in Table III. Notice the behavior of the score factor as the search window is shifted. The scene tested did not have a target present in this case.

TARGET ILLUMINATION

The grid method of normalization was next tested on scene PTANKH3. This is the same scene that was used to create template H3 (PTEMPH3). The scene was re-digitized to test the effect of the noise added in the digitization process. See Figure 21 for the digitized version of PTANKH3.

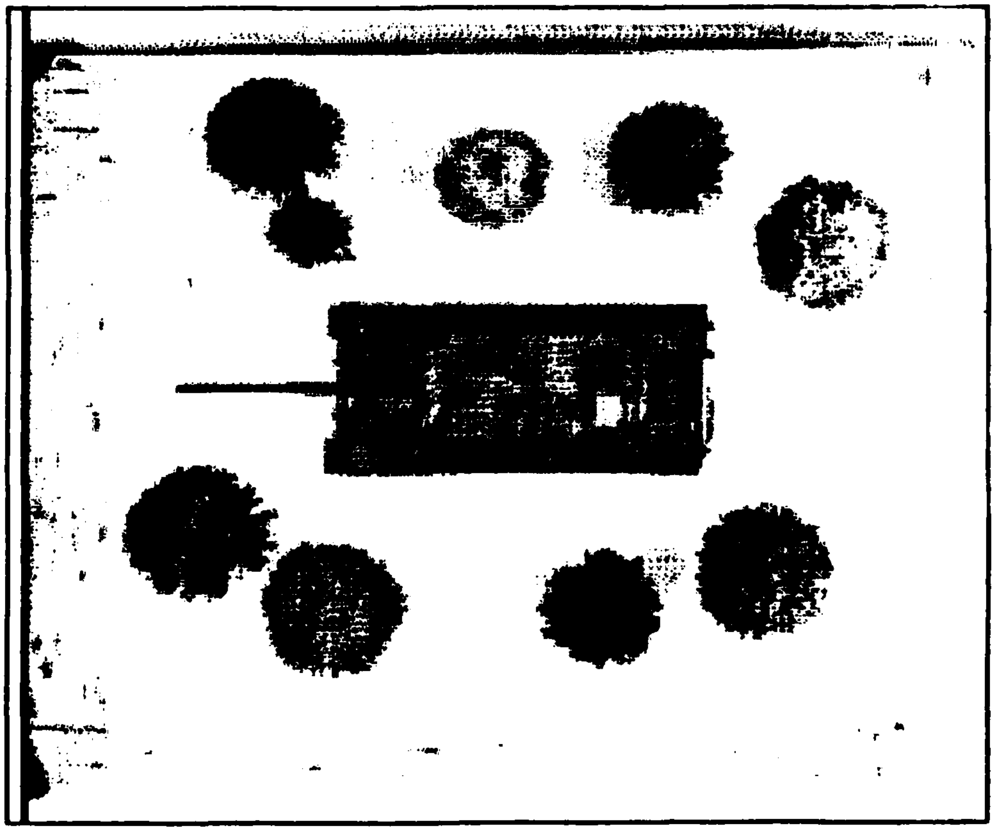
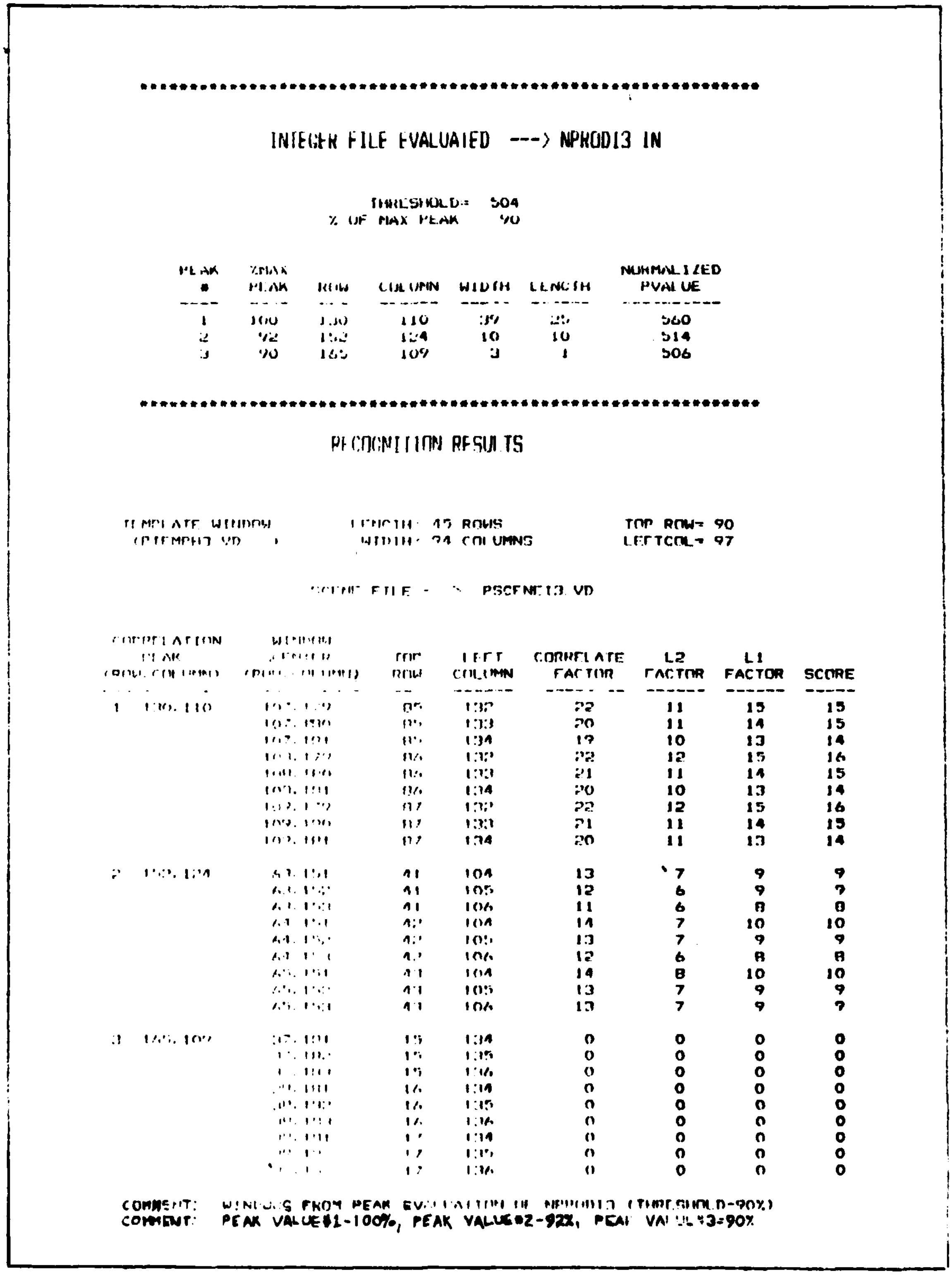


FIGURE 21: DIGITIZED TANK SCENE H3.

The CCF between PTANKH3 and PTEMPH3 is shown in Figure 22. Unfortunately, there are no distinct peaks present to

TABLE III. PEAK EVALUATION OF NPRODI3, AND CORRESPONDING DISTANCES



```
***************************************************************

              INTEGER FILE EVALUATED  ---> NPRODI3 IN

                       THRESHOLD=   504
                   % OF MAX PEAK     90

      PEAK    %MAX                                            NORMALIZED
        #     PEAK    ROW    COLUMN   WIDTH   LENGTH            PVALUE
      ----    ----    ---    ------   -----   ------          ----------
        1      100    130      110      39      25               560
        2       92    152      124      10      10               514
        3       90    165      109       3       1               506

***************************************************************

                    RECOGNITION RESULTS

 TEMPLATE WINDOW          LENGTH: 45 ROWS             TOP ROW= 90
  (PTEMPH3.VD    )         WIDTH: 24 COLUMNS          LEFTCOL= 97

                    SCENE FILE ---> PSCENE13.VD

 CORRELATION      WINDOW
    PEAK          CENTER        TOP     LEFT    CORRELATE    L2      L1
 (ROW,COLUMN)   (ROW,COLUMN)    ROW    COLUMN    FACTOR    FACTOR  FACTOR   SCORE
 ------------   ------------    ---    ------   --------   ------  ------   -----
 1   130,110      107,129        85     132        22        11      15       15
                  107,130        85     133        20        11      14       15
                  107,131        85     134        19        10      13       14
                  108,129        86     132        22        12      15       16
                  108,130        86     133        21        11      14       15
                  108,131        86     134        20        10      13       14
                  109,129        87     132        22        12      15       16
                  109,130        87     133        21        11      14       15
                  109,131        87     134        20        11      13       14

 2   152,124       63,151        41     104        13         7       9        9
                   63,152        41     105        12         6       9        9
                   63,153        41     106        11         6       8        8
                   64,151        42     104        14         7      10       10
                   64,152        42     105        13         7       9        9
                   64,153        42     106        12         6       8        8
                   65,151        43     104        14         8      10       10
                   65,152        43     105        13         7       9        9
                   65,153        43     106        13         7       9        9

 3   165,109       37,101        15     134         0         0       0        0
                   37,102        15     135         0         0       0        0
                   37,103        15     136         0         0       0        0
                   38,101        16     134         0         0       0        0
                   38,102        16     135         0         0       0        0
                   38,103        16     136         0         0       0        0
                   39,101        17     134         0         0       0        0
                 [illegible]     17     135         0         0       0        0
                 [illegible]     17     136         0         0       0        0

 COMMENT:  WINDOWS FROM PEAK EVALUATION OF NPRODI3 (THRESHOLD-90%)
 COMMENT:  PEAK VALUE#1-100%, PEAK VALUE#2-92%, PEAK VALUE#3=90%
```

suggest further investigation. Apparently the grid rectangle size was not small enough to sufficiently normalize the high energy non-target areas. For a 4x6 grid, the grid rectangle size is 30x20 pixels, for an area of 600 pixels. A finer grid may produce a more meaningful CCF for the case when the target average energy per pixel is less than the clutter average energy per pixel. The effect of varying the grid size was not studied in this thesis.

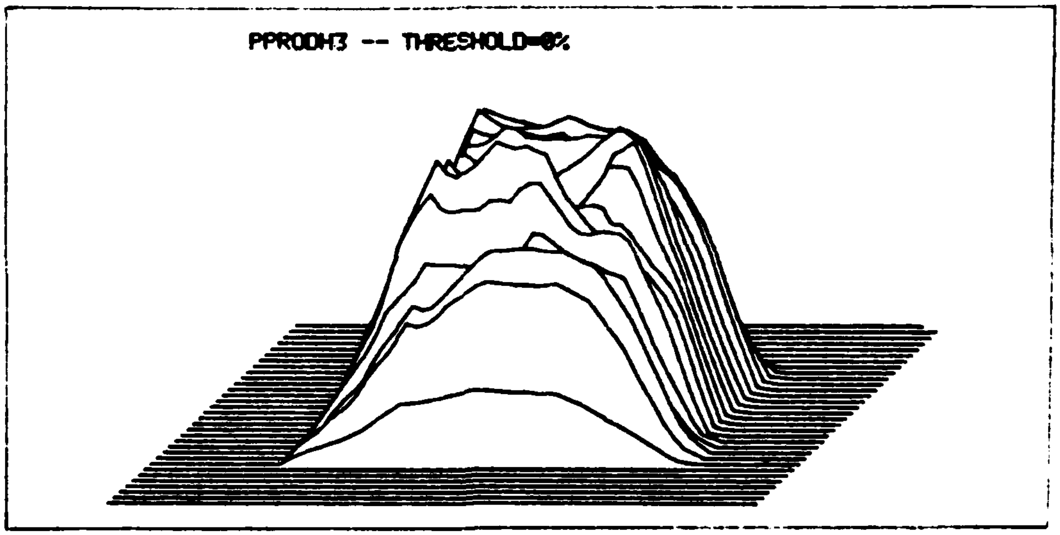


FIGURE 22: CCF OF PTANKH3 AND PTEMPH3.

The CCF of the negative of PTANKH3 (NTANKH3) and NTEMPH3 did exhibit a sharp peak, with a maximum value at (129,129). This coordinate pair corresponds to a registration shift error of only one pixel in each shift direction. See Figures 23 and 24 for the 3-D and contour plots. The peaks found, with corresponding distances, are listed in Table IV. The distance computed for Peak #1 from the reduced scene comparison indicated the best match for a shift of (128,128), as was expected. The original 256x256 scenes were compared using (128,128) as the peak location, and a distance score of 35 was computed.

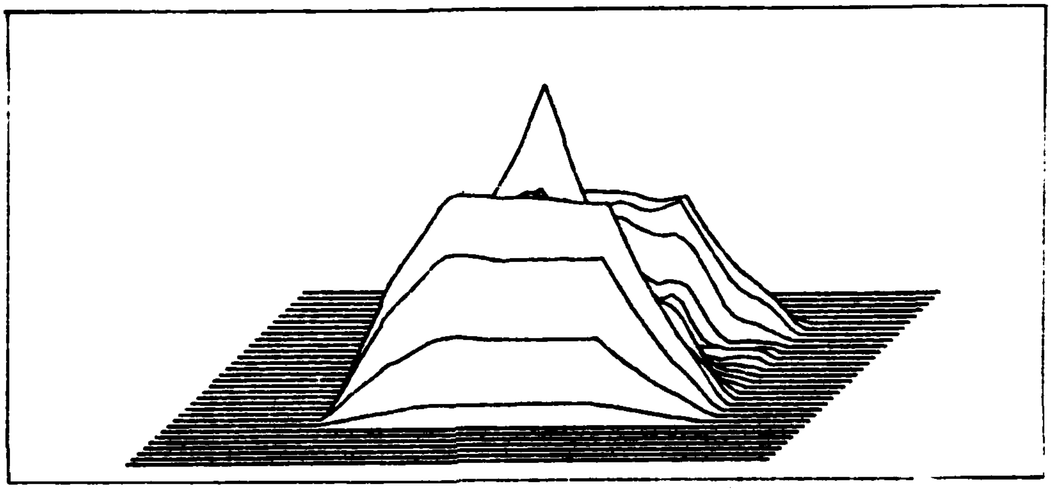


FIGURE 23: CCF OF NTANKH3 AND NTEMPH3 (NPRODH3).

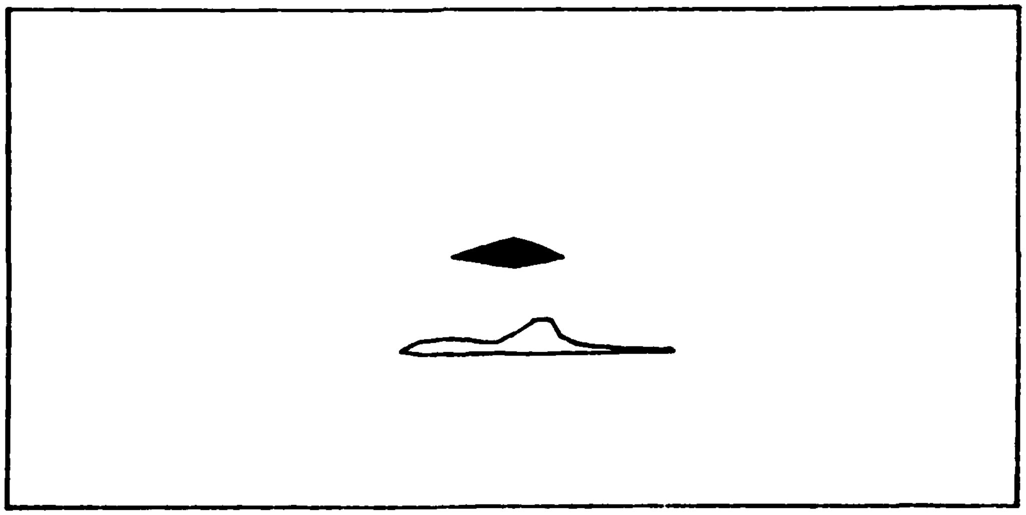
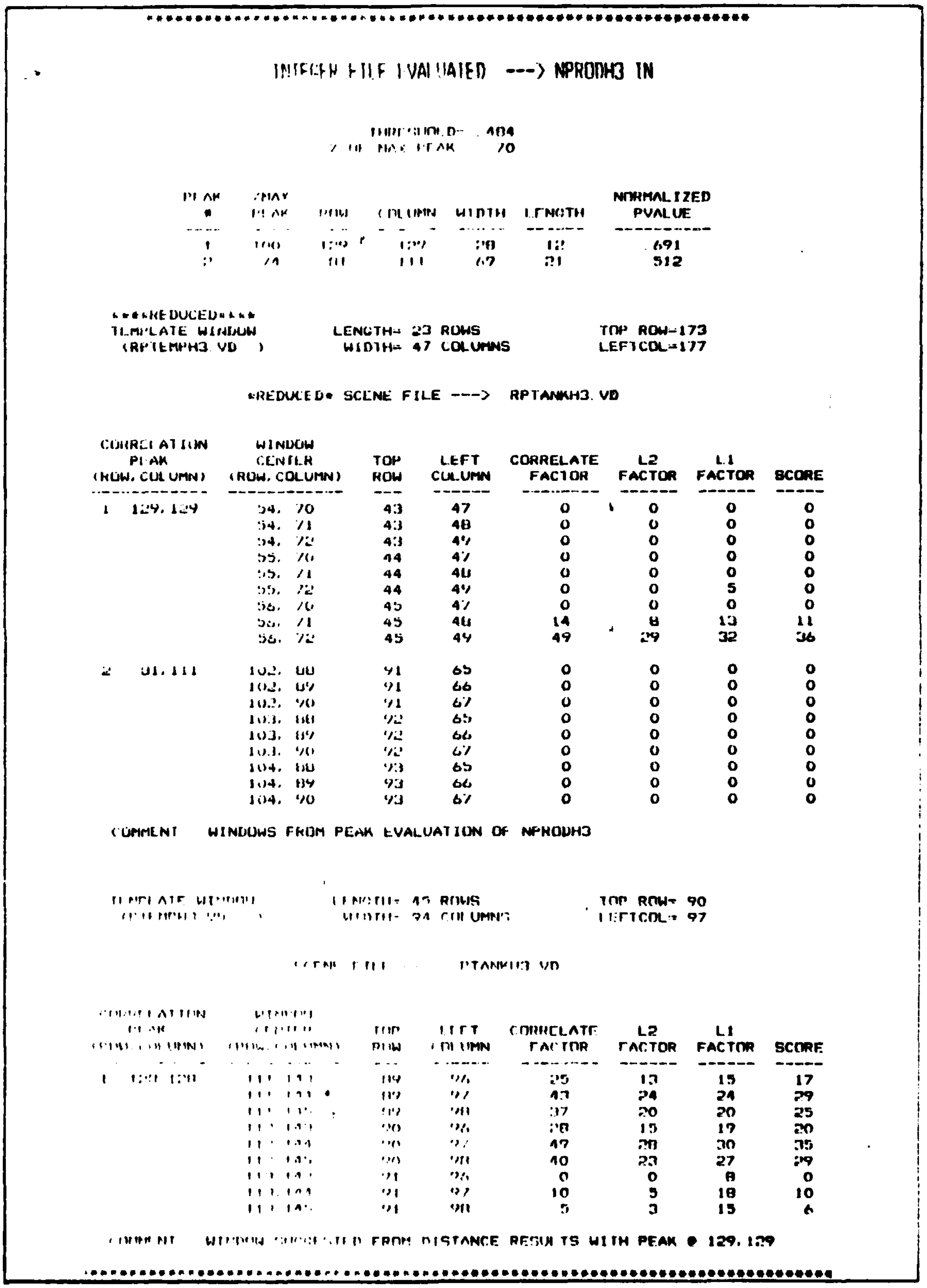


FIGURE 24: TOP 30% CONTOUR PLOT OF NPRODH3.

TABLE IV. PEAK EVALUATION OF NPRODH3, WITH CORRESPONDING DISTANCES



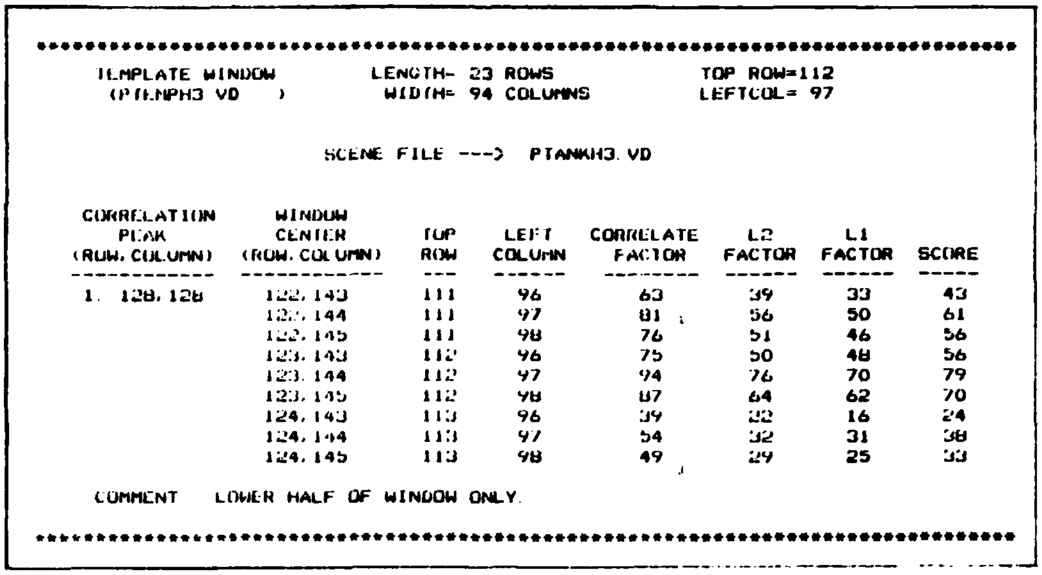
```
***********************************************************************

              INTEGER FILE EVALUATED  ---> NPRODH3 IN

                        THRESHOLD= .404
                      % OF MAX PEAK    70

      PEAK   %MAX                                           NORMALIZED
       #     PEAK    ROW    COLUMN   WIDTH   LENGTH           PVALUE
      ----   ----    ---    ------   -----   ------         ----------
       1     100     129     129      28      12               .691
       2      74      81     111      67      21               512


 ****REDUCED****
 TEMPLATE WINDOW          LENGTH= 23 ROWS            TOP ROW=173
  (RPTEMPH3.VD   )         WIDTH= 47 COLUMNS         LEFTCOL=177

                 *REDUCED* SCENE FILE --->  RPTANKH3.VD

 CORRELATION     WINDOW
    PEAK         CENTER        TOP    LEFT    CORRELATE    L2      L1
 (ROW,COLUMN)  (ROW,COLUMN)    ROW   COLUMN    FACTOR    FACTOR  FACTOR   SCORE
 ------------  ------------    ---   ------   --------   ------  ------   -----
 1  129,129       54, 70        43     47         0         0       0       0
                  54, 71        43     48         0         0       0       0
                  54, 72        43     49         0         0       0       0
                  55, 70        44     47         0         0       0       0
                  55, 71        44     48         0         0       0       0
                  55, 72        44     49         0         0       5       0
                  56, 70        45     47         0         0       0       0
                  56, 71        45     48        14         8      13      11
                  56, 72        45     49        49        29      32      36

 2    81,111     102, 88        91     65         0         0       0       0
                 102, 89        91     66         0         0       0       0
                 102, 90        91     67         0         0       0       0
                 103, 88        92     65         0         0       0       0
                 103, 89        92     66         0         0       0       0
                 103, 90        92     67         0         0       0       0
                 104, 88        93     65         0         0       0       0
                 104, 89        93     66         0         0       0       0
                 104, 90        93     67         0         0       0       0

 COMMENT   WINDOWS FROM PEAK EVALUATION OF NPRODH3


 TEMPLATE WINDOW          LENGTH= 45 ROWS            TOP ROW= 90
  (PTEMPH3.VD    )         WIDTH= 94 COLUMNS         LEFTCOL= 97

                    SCENE FILE --->  PTANKH3.VD

 CORRELATION     WINDOW
    PEAK         CENTER        TOP    LEFT    CORRELATE    L2      L1
 (ROW,COLUMN)  (ROW,COLUMN)    ROW   COLUMN    FACTOR    FACTOR  FACTOR   SCORE
 ------------  ------------    ---   ------   --------   ------  ------   -----
 1  129,129      111,143        89     96        25        13      15      17
                 111,144        89     97        43        24      24      29
                 111,145        89     98        37        20      20      25
                 112,143        90     96        28        15      17      20
                 112,144        90     97        47        28      30      35
                 112,145        90     98        40        23      27      29
                 113,143        91     96         0         0       8       0
                 113,144        91     97        10         5      18      10
                 113,145        91     98         5         3      15       6

 COMMENT   WINDOW SUGGESTED FROM DISTANCE RESULTS WITH PEAK @ 129,129
***********************************************************************
```

One of the reasons for this low score can be attributed to the method of normalization used prior to the distance calculation. As the template window and the target have been given to be the same size, the search window will contain only target information when properly placed. The clutter energy will be eliminated, so the only normalization thought to be needed was search window normalization (in other words, a 1x1 grid normalization). This type of normalization will not take into account energy changes over the target due to illumination. Recall that the brightness function is a product of the reflectance and illumination functions. It is an implicit assumption that the target reflectance function is expected to be about the same as that of the template. To take into account the variability of the illumination, both the scene and the template should be normalized by a method similar to that used in computing the CCF. Grid normalization to improve the confidence of the distance factors was not implemented due to time constraints. However, a simple demonstration of the advantage of using search window grid normalization is given in Table V. Note the increase in the best distance score factor from 35 to 79.

POSITIONING ERRORS

The correlation process was next tested using NTANKD2 as the input scene, and NTEMPH3 as the template. The results of the peak evaluation of NPRODD2 are given in Table VI. Tank

TABLE V. DISTANCE SCORES FROM COMPARISON OF LOWER HALF OF PTEMPH3 AND PTANKH3



```
********************************************************************************

   TEMPLATE WINDOW          LENGTH= 23 ROWS               TOP ROW=112
    (PTEMPH3 VD    )         WIDTH= 94 COLUMNS            LEFTCOL= 97


                         SCENE FILE ---> PTANKH3.VD


 CORRELATION       WINDOW
    PEAK           CENTER        TOP     LEFT    CORRELATE     L2       L1
 (ROW,COLUMN)   (ROW,COLUMN)     ROW    COLUMN    FACTOR     FACTOR   FACTOR   SCORE
 ------------   ------------     ---    ------   ---------   ------   ------   -----
 1.  128,128       122,143       111      96        63         39       33       43
                   122,144       111      97        81         56       50       61
                   122,145       111      98        76         51       46       56
                   123,143       112      96        75         50       48       56
                   123,144       112      97        94         76       70       79
                   123,145       112      98        87         64       62       70
                   124,143       113      96        39         22       16       24
                   124,144       113      97        54         32       31       38
                   124,145       113      98        49         29       25       33

   COMMENT    LOWER HALF OF WINDOW ONLY.

********************************************************************************
```

scene D2 and the window choices of PEAK are shown in Figures 25 and 26. The distances corresponding to the windows chosen by PEAK are given in Table VII. Notice that the first window choice clearly corresponds to the target, but that the distance factors favor the second window choice. As expected, the distance factors are very sensitive to window positioning errors, since they are based only upon a pixel-by-pixel comparison.

To achieve smaller window positioning errors, a second CCF can be computed between the template and an intermediate size window. The window size would be larger than the template, but much smaller than the original scene area. An improvement of this type would be relatively computationally inexpensive, as the FFT based correlation time is proportional to NxN log N, where N is the window width.

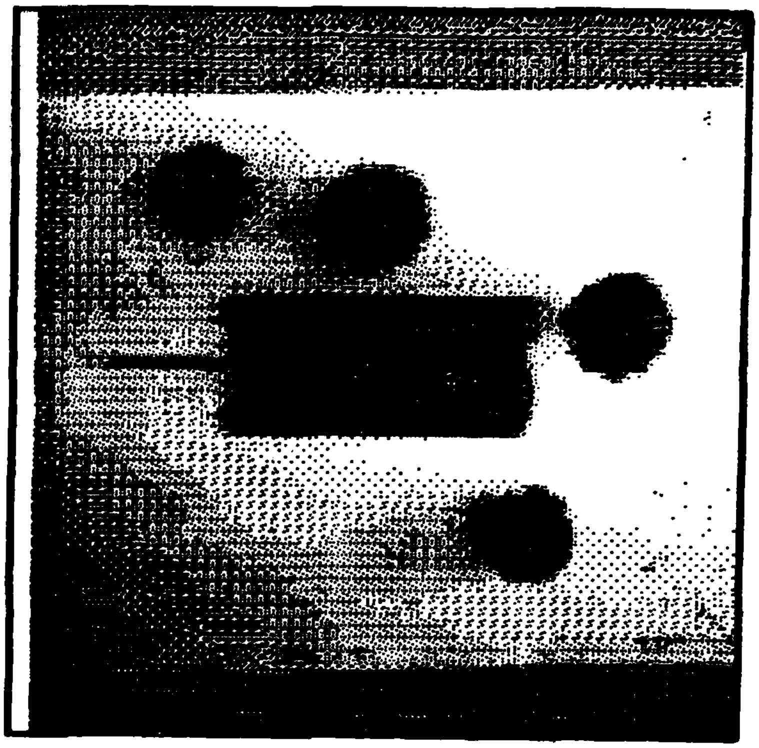


FIGURE 25: DIGITIZED TANK SCENE D2.

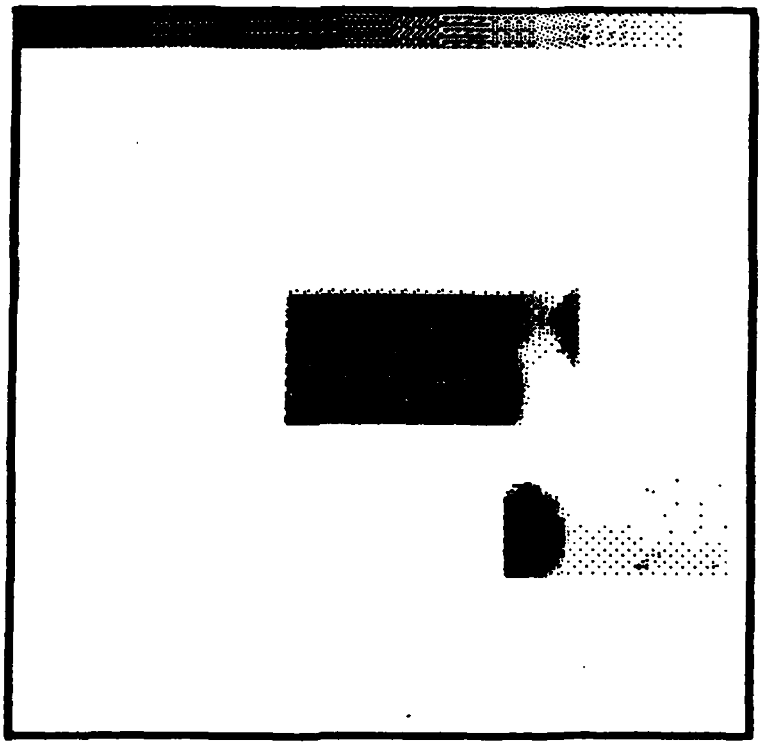
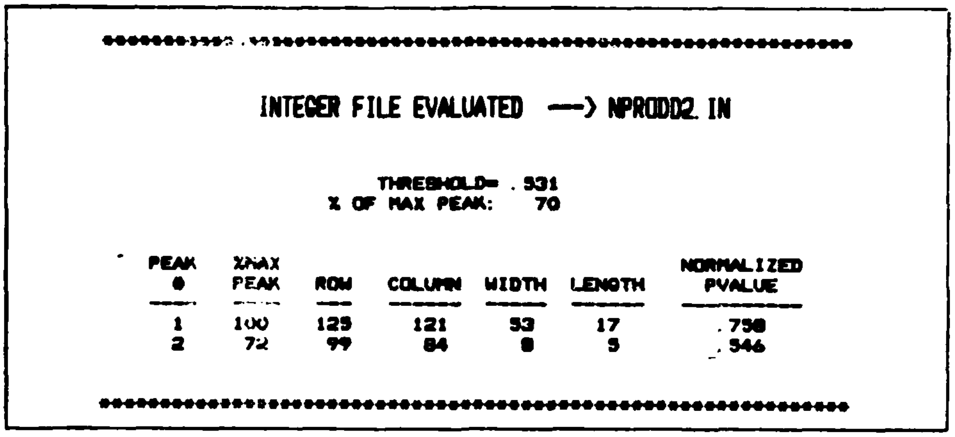


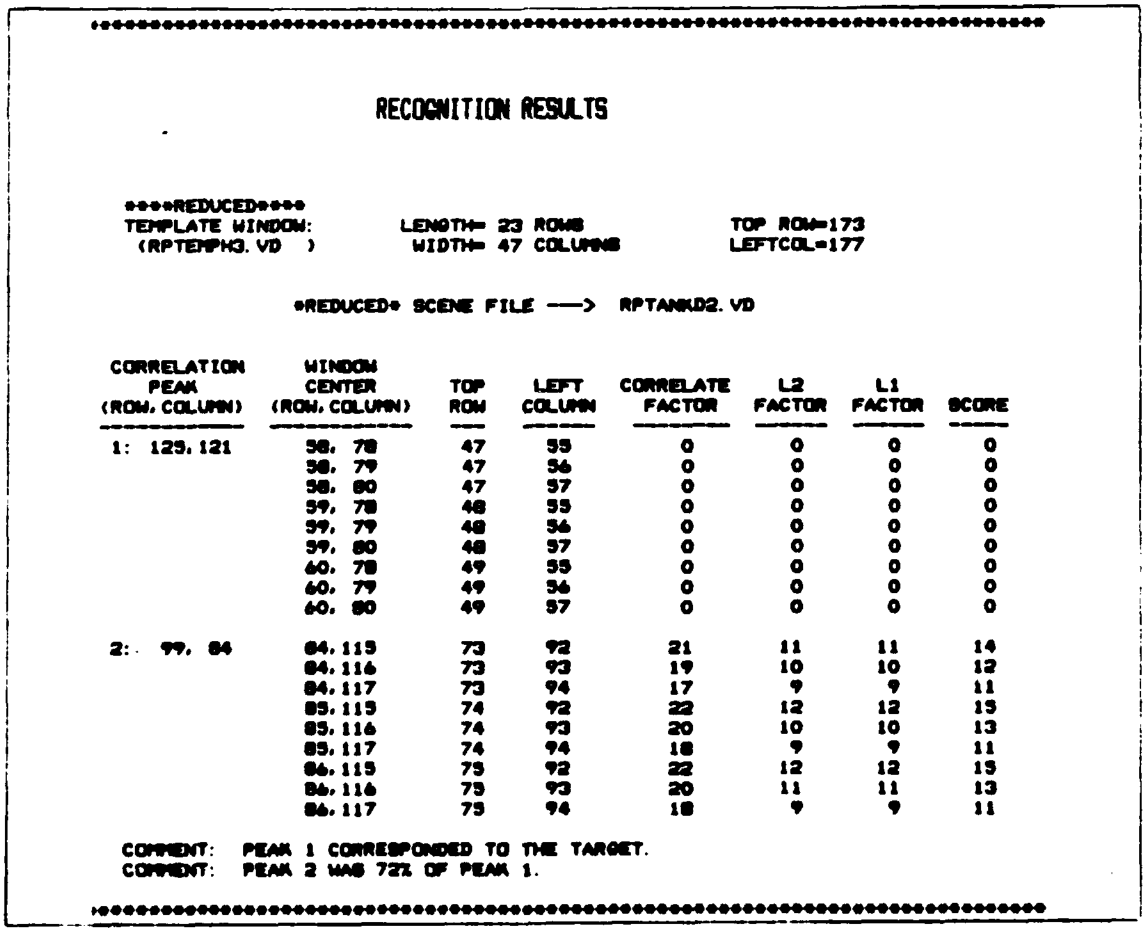
FIGURE 26: FIRST TWO WINDOW CHOICES FROM PEAK EVALUATION OF NPRODD2.

TABLE VI. PEAK EVALUATION OF NPRODD2



```
**************************************************************

            INTEGER FILE EVALUATED  ---> NPRODD2.IN

                      THRESHOLD= .531
                   % OF MAX PEAK:   70

   PEAK   %MAX                                     NORMALIZED
    #     PEAK   ROW   COLUMN  WIDTH  LENGTH         PVALUE
   ----   ----   ---   ------  -----  ------       ----------
    1     100    125    121     53     17             .758
    2      72     99     84      8      5             .546

**************************************************************
```

TABLE VII. DISTANCES CORRESPONDING TO WINDOWS OF FIGURE 26



```
*****************************************************************************

                         RECOGNITION RESULTS

 ****REDUCED****
 TEMPLATE WINDOW:          LENGTH= 23 ROWS           TOP ROW=173
  (RPTEMPH3.VD  )           WIDTH= 47 COLUMNS        LEFTCOL=177

               *REDUCED* SCENE FILE ---> RPTANKD2.VD

 CORRELATION    WINDOW
    PEAK        CENTER      TOP    LEFT   CORRELATE   L2      L1
(ROW,COLUMN)  (ROW,COLUMN)  ROW   COLUMN   FACTOR   FACTOR  FACTOR  SCORE
------------  ------------  ---   ------  --------  ------  ------  -----
 1:  125,121     58, 78      47     55        0        0       0      0
                 58, 79      47     56        0        0       0      0
                 58, 80      47     57        0        0       0      0
                 59, 78      48     55        0        0       0      0
                 59, 79      48     56        0        0       0      0
                 59, 80      48     57        0        0       0      0
                 60, 78      49     55        0        0       0      0
                 60, 79      49     56        0        0       0      0
                 60, 80      49     57        0        0       0      0

 2:   99, 84     84,115      73     92       21       11      11     14
                 84,116      73     93       19       10      10     12
                 84,117      73     94       17        9       9     11
                 85,115      74     92       22       12      12     15
                 85,116      74     93       20       10      10     13
                 85,117      74     94       18        9       9     11
                 86,115      75     92       22       12      12     15
                 86,116      75     93       20       11      11     13
                 86,117      75     94       18        9       9     11

 COMMENT:  PEAK 1 CORRESPONDED TO THE TARGET.
 COMMENT:  PEAK 2 WAS 72% OF PEAK 1.

*****************************************************************************
```

Thus, the correlation computation time quickly diminishes as N is decreased.

CLASSIFICATION ERRORS

The ability of the distance factors to classify windows was tested using PTANKD2, and the standard template, PTEMPH3. A window was selected in PTANKD2 to correspond to the tank only (the ideal choice of PEAK evaluating NPRODD2).

None of the distance factors of the nine windows investigated by DISTANCE were greater than zero, a somewhat disappointing result. There are several possible explanations for this result. The first is that scene D2 and template H3 were digitized in separate sessions. As a result, the size and orientation of the tanks will slightly differ. The second explanation is that the normalization approach used in the distance calculation admittedly does not account for illumination changes over the target, as mentioned previously. Finally, the 16 level quantization cannot accurately represent the continuous image unless the brightness function histogram is reasonably "spread out."

The third explanation requires further discussion. Consider the histograms of template H3, and of the tank of scene D2 (Figures 27 and 28). The histogram of template H3 is almost evenly distributed from levels 0 to 9. On the other hand, the nistogram of the tank of scene D2 shows 63%

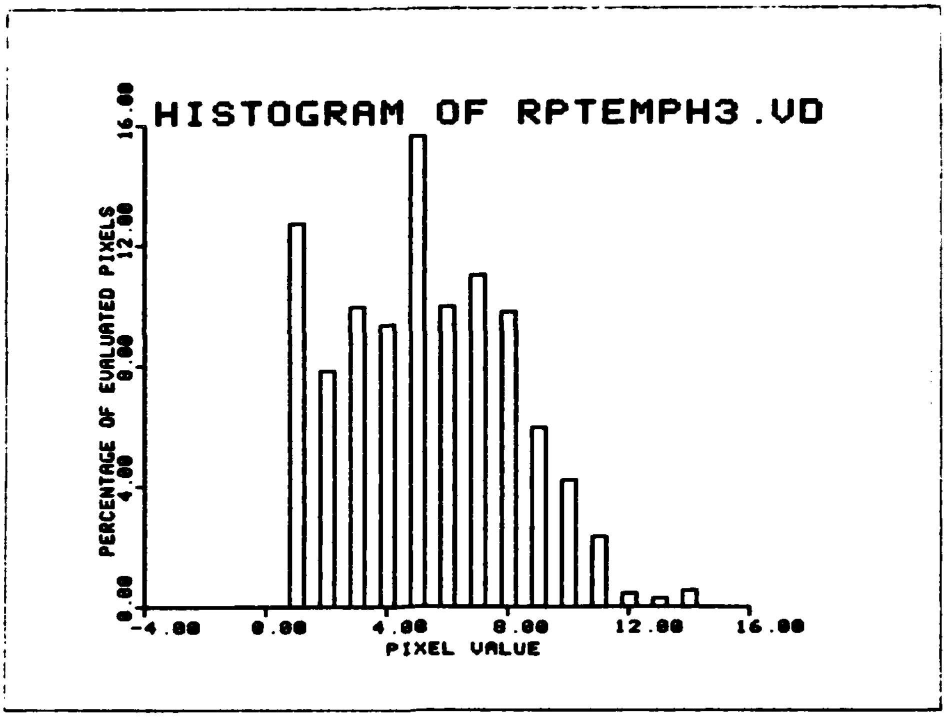


FIGURE 27: HISTOGRAM OF TEMPLATE H3.

of the pixels to be in levels 2 and 3. This "distortion" from the ideal (template) histogram will result in some loss of detail in the digitized image. Recall that one of the assumptions made is that the digitized images must be accurate representations of the continuous scene, or dependable detection cannot be expected.

The histogram of RTEMPD2 was modified by TONER so that any detail in the scene would be enhanced. The pixel mapping is shown in Table VIII, and the resulting histogram is given in Figure 29. The enhanced version of TANKD2 is compared with template H3 in Figure 30. The effect of the difference in illumination can clearly be seen by comparing the top

HISTOGRAM OF RTEMPD2.VD

```
# of level 0:     0        # of level  8:    16
# of level 1:    24        # of level  9:    13
# of level 2:   353        # of level 10:     3
# of level 3:   294        # of level 11:     5
# of level 4:   135        # of level 12:     6
# of level 5:    82        # of level 13:    35
# of level 6:    31        # of level 14:    21
# of level 7:    14        # of level 15: 64504
```

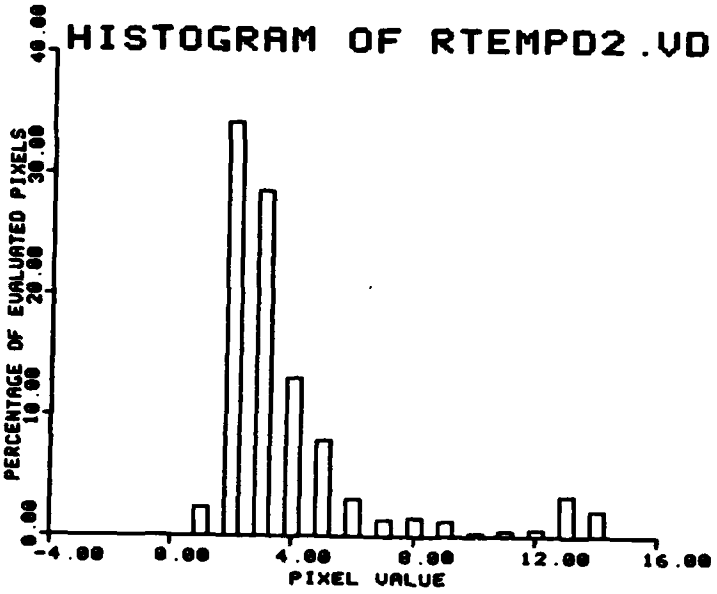
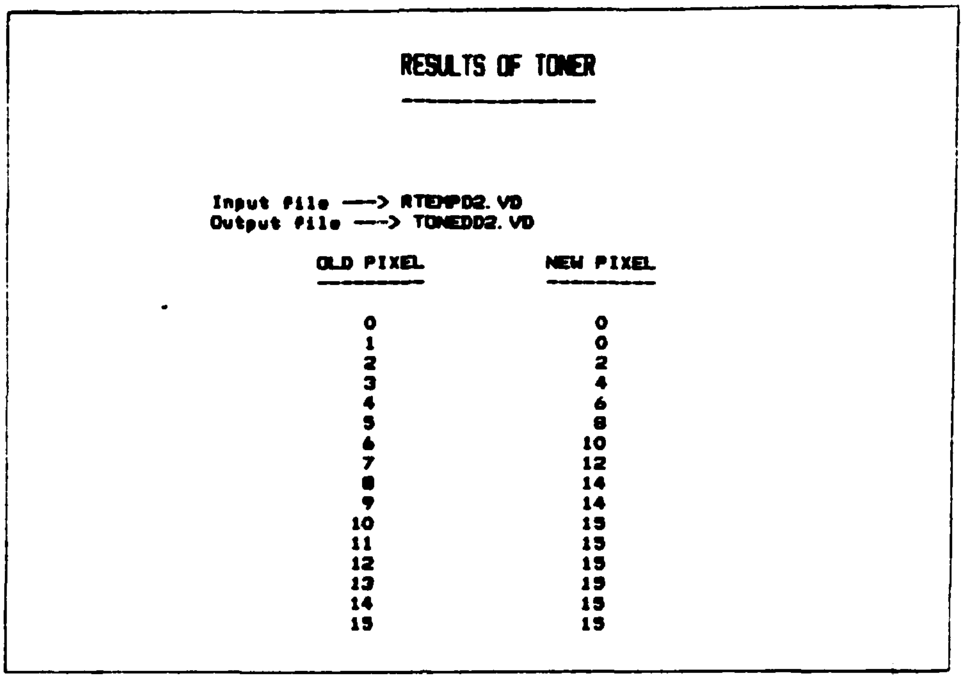


FIGURE 28: HISTOGRAM OF TANK D2
(BACKGROUND PIXEL VALUES SUPPRESSED).

portions of the tanks, and also the bottom portions. Unless an elegant method of normalization is used, any point-by-point based distance measure may give misleading results.

There are several methods of computing distances that take into account the spatial relationships of the pixels in an image. Although none were implemented in this study, they

TABLE VIII. PIXEL MAPPING FOR TONED SCENE D2



```
                    RESULTS OF TONER
                    ----------------

Input file ---> RTEMPD2.VD
Output file ---> TONEDD2.VD

           OLD PIXEL         NEW PIXEL
           ---------         ---------

               0                 0
               1                 0
               2                 2
               3                 4
               4                 6
               5                 8
               6                10
               7                12
               8                14
               9                14
              10                15
              11                15
              12                15
              13                15
              14                15
              15                15
```

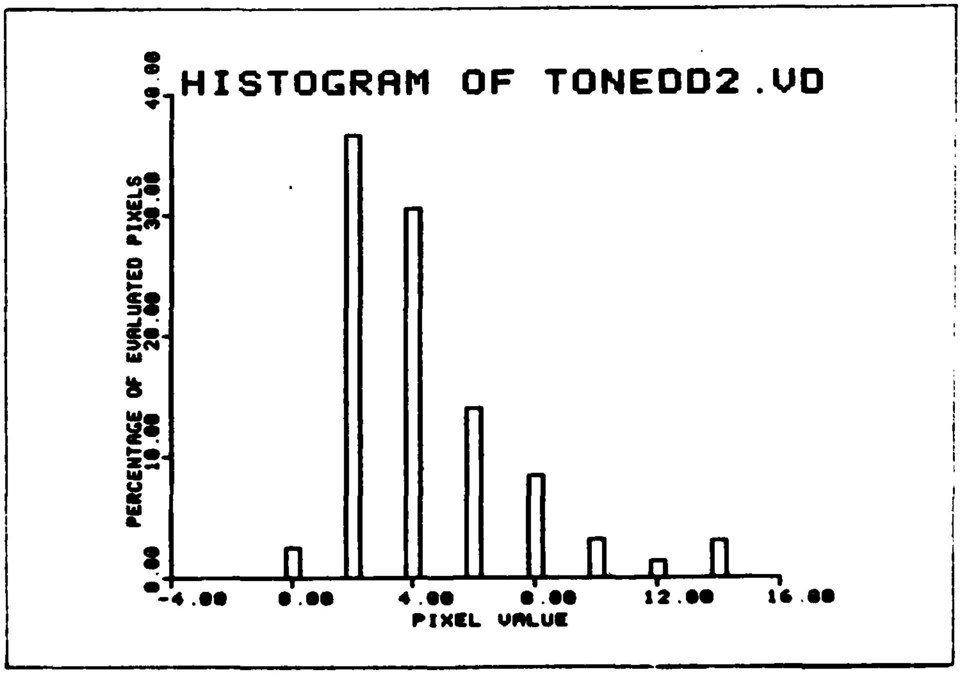


FIGURE 29: HISTOGRAM OF TONEDD2.

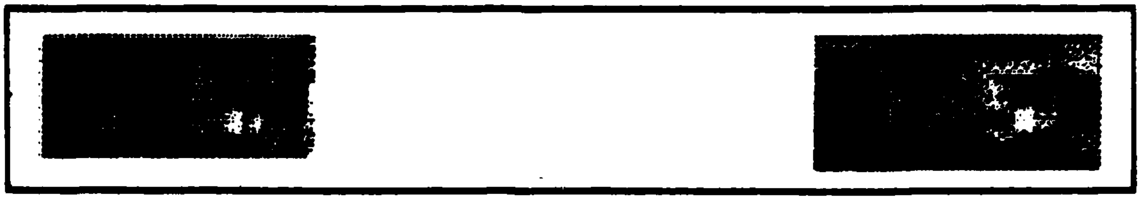


FIGURE 30: TONED TANK D2 (LEFT) AND TEMPLATE H3 (RIGHT).

are suggested as logical improvements to the detection process. One method to be tested is to compare in some way the filtered Fourier components of the normalized scene window with those of the normalized template window. Another possibility to explore is to compute a statistical correlation measure of the form

$$Ng(m,n) = \frac{\sum_x \sum_y f\{s(x,y)\} f\{t(x-m,y-n)\}}{\sqrt{\sum_x \sum_y [f\{s(x,y)\}]^2 \sum_x \sum_y [f\{t(x-m,y-n)\}]^2}} \qquad (49)$$

where f{ • } indicates some 3x3 mask operation of the array, most likely an edge enhancer. Linear edge enhancers include the Laplacian mask, while non-linear enhancers include the Kirsch and Roberts operators [5: 482-491]. A 3x3 mask operator can easily be implemented by modifying the subroutine TEST3 of the program REMOVE.

VII. CONCLUSION

SUMMARY

In this thesis, a method for two-dimensional pattern recognition was developed and tested. The method included a global search scheme for candidate targets, based on high speed cross-correlation between a normalized scene and template. A target classification measure dependent on the normalized L1 and Euclidean distances was also presented.

Several computer programs were written to carry out the process, from image input to target classification. Especially significant was the program PEAK, which performs a search of two-dimensional cross-correlation functions for isolated supra-threshold peaks. Another program, DISTANCE, computes a classification, or similarity score between template and scene windows corresponding to the correlation peaks. Simple image procesing programs were also written and described.

CONCLUSIONS

1. The global search for candidate targets by cross-correlation template matching can be counted on to find candidate targets to only a limited degree. The search scheme presented in this study is best used to determine the approximate location of amorphous "blobs."

2. The reliability of cross-correlation in finding targets increases as the target-to-clutter area ratio increases.

This property can allow the target to be found by an iterative process of correlationss using smaller and smaller window sizes.

3. Similarity measures based on pixel-by-pixel comparisons are sensitive to slight mis-registration errors, and to minor degradations in the brightness function due to illumination and digitization noise.

4. The grid normalization method of chapter four for use in the correlation process is not a suitable approximation of search window normalization. The method has difficulties when a grid rectangle covers an area which contains both high and low energy objects. However, it works very well for normalizing scenes and templates in the classification process, after a search window has been chosen.

RECOMMENDATIONS

1. An algorithm to automatically evaluate the information generated by the program PEAK needs to be developed. Specifically, a simple rule to determine the selection of a threshold value (or possibly several values for a given CCF) needs to be determined. Also to be determined is a method to recognize a false peak (one not corresponding to a target), given its width, length, and percentage of maximum peak value. Perhaps the cross-section of the peak could be compared with the cross-section of the autocorrelation function. Elimination of false peaks from further consideration would result in computational savings in the

classification process.

2. A more elegant normalization scheme is needed for use in the global search process. One possibility would be to compute a finer array of normalization constants that take into account the energies of the surrounding grid rectangles (see Figure 31). The normalization coefficients computed would essentially be the result of overlapping the grid rectangles.

3. An intermediate correlation using a smaller window area should be computed, using the peak information from the global search CCF to determine the window size and center. This intermediate correlation will allow for a precise positioning of the window used in the computation of the similarity measure.

4. The use of grid normalization of scene and template windows for the classification process needs to be further developed and tested. Preliminary results indicate that grid normalization can be used successfully to account for differences in energy between the template and scene tanks, as long as most of the clutter energy is eliminated (which is the case when the search window is accurately positioned).

5. The choice of features to be compared needs to be studied. Instead of comparing the pixel values of the scene candidate window directly with those of template window, the comparison of the high or low-frequency Fourier components, for example, may lead to more promising results. It might also be interesting to compute the distances between scenes

FIGURE 31: SUGGESTED NORMALIZATION SCHEME

Consider the case where the energies of 9 local rectangular "whole" regions A-I are computed.

A	B	C
D	E	F
G	H	I

From these 9 energy terms, we want to compute 81 normalization coefficients. The coefficients for any "sub"-region can be approximated by a weighted average of the whole region coefficients corresponding to the center sub-region and its eight nearest neighbors, where the weights are determined by the distance from the center sub-region.

A	A	B
A	A'	B
D	D	E

For s(x,y) in sub-region A', one possible method of computing a normalization coefficient is by Eq. 50 (let N(A') represent the normalization coefficient for region A'):

$$N(A') = \sqrt{K1xA + K2x\left[(A + A + B + D) + (A + B + E + D)/\sqrt{2}\right.}\left.\right] \quad (50)$$

This normalization method may avoid some of the severe discontinuities sometimes created by whole region normalization.

and templates that have been mask processed (for example edge enhanced), as mentioned in chapter six.

BIBLIOGRAPHY

1. Casasent, David. "Pattern Recognition: A Review," IEEE Spectrum, 18: 28-33 (March 1981).

2. Rosenfield, Azriel. "Image Pattern Recognition," Proceedings of the IEEE, 69: 596-605 (May 1981).

3. Horev, Moshe. "Picture Correlation Model for Automatic Machine Recognition." Unpublished MS thesis. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, December 1980.

4. Duda, Richard O. and Peter E. Hart. Pattern Classification and Scene Analysis. New York: John Wiley and Sons, Inc., 1973.

5. Pratt, William K. Digital Image Processing. New York: John Wiley and Sons, Inc., 1978.

6. Bracewell, Ronald N. The Fourier Transform and Its Applications (Second Edition). New York: McGraw-Hill, Inc., 1978.

7. Oppenheim, Alan V. and Ronald W Schafer. Digital Signal Processing. Englewood Cliffs, NJ: Prentice-Hall, 1975.

8. Fredal, Dan and George C. Beasley, Jr. "An Analog Speech I/O Channel for the Nova 2 Computer Based on the S-100 Bus." MS thesis. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, March 1981. (AD A103 398)

9. Simmons, Robin A. "Machine Segmentation of Unformatted Characters." MS thesis. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, December 1981. (AD A115 556)

APPENDIX A:

USE OF MACRO FILES

Under the RDOS operating system, macro (or indirect files) can be run to execute a series of CLI commands. Thus, a process consisting of a series of program can be run automatically as long as none of the programs require interactive user input. In order that file names and program options can be specified in the execution line, all of the programs in the correlation process use the COMARG call to read the command line argument string. This appendix describes the use of several macro programs.

NTEMPCP.MC

Macro NTEMPCP.MC creates NTEMP.CP, the Fourier transform of the reduced template, RNTEMP.VD. RNTEMP.VD is the negative of the 256x256 PTEMP.VD, reduced to the lower right quadrant by REDUCE and NMOVE. Links to NORMALIZE and DIRECT must exist before NTEMPCP.MC is executed. The program lines are as follows:

```
NORMALIZE/L RNTEMP.VD NTEMP.CP
DELETE/V RNTEMP.VD
DIRECT NTEMP.CP/I 256/N
```

NPRODIN.MC

Execution of NPRODIN.MC results in the CCF between the template and SCENE.VD to be computed and stored in NPROD.IN. Files HOLD1.CP and HOLD2.CP should exist as 1024-block contiguous files; they will be created if they do not exist. The use of contiguous (as opposed random) files decreases run time by a factor of 3. The following files must be linked to before NPRODIN.MC can be run (the links in CROMER.DR are given as an example):

```
 CROMER
:26:26

HOLD1.CP                  STROUPE:HOLD1.CP
HOLD2.CP                  STROUPE:HOLD2.CP
NTEMP.CP                  STROUPE:NTEMP.CP
SCENE.VD                  STROUPE:SCENE.VD
NEGATE.SV                 STROUPE:NEGATE.SV
REDUCE.SV                 STROUPE:REDUCE.SV
NORMALIZE.SV              STROUPE:NORMALIZE.SV
DIRECT.SV                 DP4F:DIRECT.SV
CMULTIPLY.SV              STROUPE:CMULTIPLY.SV
INVERSE.SV                DP4F:INVERSE.SV
CTOI.SV                   STROUPE:CTOI.SV
```

Run time will be less than 10 minutes. The scene of interest should be renamed SCENE.VD before executing; NPROD.IN should be renamed after the macro program execution. See COMPUTE.MC for further clarification on the use of NPRODIN.MC. One version of NPRODIN.MC is as follows:

```
RENAME HOLD1.CP NSCENE.CP
RENAME HOLD2.CP NPROD.CP
NEGATE SCENE.VD NSCENE.VD
REDUCE NSCENE.VD RNSCENE.VD
DELETE/V NSCENE.VD
```

```
NORMALIZE/U RNSCENE.VD NSCENE.CP
DELETE/V RNSCENE.VD
DIRECT NSCENE.CP/I 256/N
CMULTIPLY NTEMP.CP NSCENE.CP NPROD.CP
RENAME NSCENE.CP HOLD1.CP
INVERSE NPROD.CP/I 256/N
CTOI NPROD.CP NPROD.IN
RENAME NPROD.CP HOLD2.CP
```

COMPUTE.MC

COMPUTE.MC is used to automatically, without user interaction, compute the CCF's corresponding to six different scenes and one template. The total run time is less than one hour. The macro controls the re-naming (or re-linking in this case) of the dummy files SCENE.VD and NPROD.IN. The CCF's created are moved to other disks to allow room to run the rest of the correlations. The macro program requires 256K bytes of free disk (assuming NPROD.IN exists). If for any reason NPROD.IN cannot be moved, it is simply overwritten. The program may be aborted at any time the user desires; results up to that time will be saved. The program is given as follows:

```
LINK SCENE.VD CROMER:PTANKG4.VD
NPRODIN
RENAME NPROD.IN NPRODG4.IN
UNLINK SCENE.VD
LINK SCENE.VD CROMER:PTANKD4.VD
NPRODIN
MOVE/V/D/R DP5F NPROD.IN/S NPRODD4.IN
LINK NPRODD4.IN DP5F:NPRODD4.IN
UNLINK SCENE.VD
LINK SCENE.VD CROMER:PTANKC3.VD
NPRODIN
MOVE/V/D/R DP5F NPROD.IN/S NPRODC3.IN
```

```
LINK NPRODC3.IN DP5F:NPRODC3.IN
UNLINK SCENE.VD
LINK SCENE.VD CROMER:PTANKB2.VD
NPRODIN
MOVE/V/D/R DP5F NPROD.IN/S NPRODB2.IN
LINK NPRODB2.IN DP5F:NPRODB2.IN
UNLINK SCENE.VD
LINK SCENE.VD CROMER:PTANKD2.VD
NPRODIN
MOVE/V/D/R DP5 NPROD.IN/S NPRODD2.IN
LINK NPRODD2.IN DP5:NPRODD2.IN
UNLINK SCENE.VD
LINK SCENE.VD CROMER:PTANKE2.VD
NPRODIN
MOVE/V/D/R DP5 NPROD.IN/S NPRODE2.IN
LINK NPRODE2.IN DP5:NPRODE2.IN
UNLINK SCENE.VD
DIR DP4
MESSAGE RECOGNITION PROGRAM COMPLETED
GDIR
GTOD
```

The scene files correlated with the template are as follows:

```
    CROMER


PTANKB2.VD          32768   PC
PTANKC3.VD          32768   PC
PTANKD2.VD          32768   PC
PTANKD4.VD          32768   PC
PTANKE2.VD          32768   PC
PTANKG4.VD          32768   PC
```

The files and links created by this version of COMPUTE.MC are as follows:

```
  CROMER


NPRODB2.IN                     DP5F:NPRODB2.IN
NPRODC3.IN                     DP5F:NPRODC3.IN
NPRODD2.IN                     DP5:NPRODD2.IN
NPRODD4.IN                     DP5F:NPRODD4.IN
NPRODE2.IN                     DP5:NPRODE2.IN
NPRODG4.IN        131072   C
```

APPENDIX B:
SUMMARY OF PROGRAM USAGE

INPUT FILE TYPE	EXECUTION LINE FORMAT	OUTPUT FILE TYPE
[.VD]	VIDEO7	[.VD]
.VD	QUICKAVE7	AVERAGE7.VD
.VD	REMOVE infile out1 out2	.VD
.VD	EVIDHIST	—
.VD	TONER	.VD
.VD	NMOVE	.VD
.VD	NEGATE[/F] infile outfile	.VD
.VD	REDUCE infile outfile	.VD
.VD	NORMALIZE[/U or /L] infile outfile	.CP
.CP	DIRECT infile/I 256/N [outfile/O]	.CP
.CP	CMULTIPLY infile1 infile2 outfile	.CP
.CP	INVERSE infile/I 256/N [outfile/O]	.CP
.CP	CTOI infile outfile	.IN
.IN	IMULTIPLY infile1 infile2 outfile	.IN
.IN	ITOC/[A,N,E,H or O] infile[/C][/T] outfile[/M]	.CP
.CP	PLTTRNS infile/I 256/N	—
.IN	PEAK	—
.VD	DISTANCE	—
.CP	CTOV	.VD
.VD	PICTURE	—

where

.VD — 32K bytes packed video

.IN — 128K bytes integer

.CP — 512K bytes complex

[ ] — optional input or output

APPENDIX C:

IMAGE INPUT AND OUTPUT PROGRAMS

This appendix contains the following programs (subroutines given in parenthesis):

1. VIDEO7 (CHAN7, VABORT, ERCHK)
2. QUICKAVE7
3. PICTURE (OUT3X3, OUT4X3)

```
C****** NOVA VIDEO7 INPUT/OUTPUT ROUTINE VIA CROMEMCO COMPUTER  *****
C                                                              *****
C      WRITTEN BY Lt. Jim Cromer            12 Aug 1982        *****
C      Fortran IV                                              *****
C                                                              *****
C      This program allows the user to display a video file    *****
C      repeatedly any number of times.  It also allows the     *****
C      user to input or output video files named A0, A1, A2,   *****
C      A3, A4, A5, etc. automatically.  These files may        *****
C      then be averaged by QUICKAVE7.                          *****
C                                                              *****
C      Execution Line Format:                                  *****
C             VIDEO7                                           *****
C                                                              *****
C      Load Line Format:                                       *****
C             RLDR VIDEO7 CHAN7 ERCHK VABORT CHANNEL           *****
C                    DCHRX DCHTX SANDS CANDR FORT.LB           *****
C                                                              *****
C*******************************************************************
      DIMENSION IPAR(2),IHOLD(7)
      INTEGER FILE(7)
      IPAR(2)=0
C
C
C******** USER PARAMETER INPUT *************************************
C
      TYPE"NOTICE:  CHOPS must be running!<12>"
      ACCEPT"Input or output (IN-0/OUT-1)? ",IDIR
      IF(IDIR.NE.0.AND.IDIR.NE.1)GO TO 8  ;Error checking
 1    ACCEPT"Enter time (SEC.):  ",ITIME  ;Monitor display time
      IF(IDIR.EQ.0)GO TO 4
      ACCEPT"Type 1 to output A0-An:  ",IDIR
      IF(IDIR.EQ.1)GO TO 4
      ACCEPT"What is the name of the data file (13 Char max)? "
      READ(11,8)FILE(1)                  ;Video frame filename
      ACCEPT"Enter # times to be displayed: ",II
      IF(FILE(1).NE.10752)GO TO 42        ;An "*" means to
      DO 2 I=1,7                          ;run with the last
   2  FILE(I)=IHOLD(I)                    ;file used
      GO TO 42
C
C
C******** INPUT OR OUTPUT FILES A0-A6 ******************************
C
   4  IPAR(1)=ITIME
      CALL CHAN7(IDIR,IPAR)
      GO TO 6
C
C
C******** DISPLAY VIDEO FILE OF USER'S CHOICE **********************
C
  42  DO 43 KK=1,II
      TYPE"CHECK MONITOR - - ",KK
      CALL VABORT(NTSK,IM,IPCNT,IDCNT)
```

```
        IDIR=1
        IPAR(1)=ITIME
        IERR=0
        CALL CHANNEL(NTSK,IDIR,IM,IPCNT,IDCNT,FILE,64,0,IPAR,IERR,ISYS)
 43     CALL ERCHK(IERR,IDIR,IDCNT,IPCNT,ISYS)  ;identify errors
C
C
C******** USER INPUT ******************************************
C
  6     TYPE"<12>"
        DO 7 I=1,7                                    ;Save filename
  7     IHOLD(I)=FILE(I)                              ;for next loop
        ACCEPT"What next (INPUT-0,OUTPUT-1,STOP-2)? ",IDIR
        IF(IDIR.EQ.0)GO TO 1
        IF(IDIR.EQ.1)GO TO 1
        IF(IDIR.EQ.2)STOP "Type VIDEO7 to rerun."
  8     FORMAT(S13)                             ;Filename input format
        TYPE"<7>Input error; try again."
        GO TO 6
        END
C
C************ Program VIDEO7 ************************************
```

```
      SUBROUTINE CHAN7(IB,IPAR)        ;by Lt Jim Cromer
C*************************************************************************
C
C     Subroutine CHAN7 will digitize a picture seven times, and
C     will store the video data in files A0-A6 (i.e. in a format
C     usable by QUICKAVE7).  It will also display the digitized
C     pictures consecutively on the video monitor.
C     (Called by VIDEO7)
C
C     PARAMETERS PASSED:
C
C             IB=0   ---> input pictures from camera
C             IB=1   ---> output pictures to the video monitor
C             IPAR(1) --> display time in seconds
C             IPAR(2) --> unused
C
C*************************************************************************
      INTEGER FILE(7),IPAR(2)
      DO 100 I=1,7
 100    FILE(I)=0
      IERR=0
      IF(IB.EQ.1)GO TO 300
      DO 200 I=16688,16694     ;if digitizing, delete A0-A6
              FILE(1)=I
 200          CALL DELETE(FILE)
      INUM=7
      GO TO 400
 300  TYPE"Enter the number of pictures to be displayed."
      TYPE"They must be named A0-A(N-1) to be displayed."
      ACCEPT"      Enter N --->  ",INUM
      IF(INUM.GT.10)TYPE"Sorry. The maximum number of pictures
     $ is 10."
      IF(INUM.GT.10)INUM=10
 400  DO 500 I=1,INUM          ;if I=1, then FILE="A0"
                               ;if I=2, then FILE="A1"
              FILE(1)=16687+I ;and so on
              CALL VABORT(IA,IC,ID,IE)
              WRITE(10,600)FILE(1)
              CALL CHANNEL(IA,IB,IC,ID,IE,FILE,64,0,IPAR,IERR,ISYS)
 500          CALL ERCHK(IERR,IB,IE,ID,ISYS)
 600  FORMAT("  Check monitor -- Picture being displayed ---> ",S13)
      RETURN
      END
C
C*********** Subroutine CHAN7 ******************************************
```

```
        SUBROUTINE VABORT(NTSK,IM,IPCNT,IDCNT)
C*****************************************************************
C
C       This subroutine sends an abort call to CHANNEL,
C       and resets the calling parameters for the next
C       VIDEO7 call.
C
C*****************************************************************
        J=0             ;dummy variable
        K=3             ;mode 3 ---> Abort
        CALL CHANNEL(J,J,K,J,J,"A",J,J,J,IE,IS)
        IDCNT=4
        IPCNT=1         ;parameter count
        NTSK=3
        IM=2            ;mode 2 --->  Video
        RETURN
        END
C
C************* Subroutine VABORT *********************************
```

```
      SUBROUTINE ERCHK(IERR,IDIR,IDCNT,IPCNT,ISYS)
C**********************************************************************
C
C     This subroutine checks for errors made during an attempted
C     call to CHANNEL, and prints them to the screen.
C     (Called by VIDEO7).
C
C**********************************************************************
      INTEGER CROERR,PDCONT
      LOGICAL BTEST
      IF(IERR.EQ.0.OR.(IERR.EQ.13323.AND.IDIR.EQ.0))GO TO 6      ;ok
      IF(IERR.EQ.13323.OR.IERR.EQ.-24832.OR.IERR.EQ.-24064.OR.IERR
     *.EQ.-22528)GO TO 5        ;Specific error messages will be given
      IF(BTEST(IERR,15))GO TO 9                    ;Abnormal return
      GO TO 10                                     ;Error without abort
    5 IF(IERR.EQ.-24832)TYPE"<7>ABORT—FILE DOES NOT EXIST"
      IF(IERR.EQ.13323.OR.IERR.EQ.-24064)TYPE"<7>ABORT—FILE DOES NOT
     * CONTAIN VIDEO"
      IF(IERR.EQ.-22528)TYPE"<7>ABORT—FILE CANNOT BE CREATED"
   6  TYPE"<12>Channel cleared"
      GO TO 20
    9 TYPE"<7><12> ABORT INITIATED!!!<12>"          ;There are
   10 CROERR=15.AND.IERR                            ;two error
      PDCONT=ISHFT(240.AND.IERR,-4)                 ;codes in the
      NOVERR=ISHFT(-256.AND.IERR,-8)                ;variable
      TYPE" CROMEMCO ERROR RETURNED:",CROERR        ;'IERR'
      TYPE" PARAMETER/DATA COUNT RETURNED:",PDCONT
      CALL BCLR(NOVERR,7)
      TYPE" NOVA ERROR RETURNED:",NOVERR            ;Error
      TYPE" ERROR CODE RETURNED:",IERR              ;messages are
      TYPE" DATA COUNT:",IDCNT                      ;printed for
      TYPE" PARAMETER COUNT:",IPCNT                 ;user information
      TYPE" SYSERR RETURNED:",ISYS                  ;and correction
      PAUSE
  20  RETURN
      END
C
C************** Subroutine ERCHK ***************************************
```

```
C***********************************************************************
C
C      Program QUICKAVE7        Written by Lt. Jim Cromer
C      Fortran 5 (by DATA GENERAL)
C
C
C            This program will average the packed video files
C      "A0" - "A6" pixel by pixel, and output the averaged picture
C      to the packed video array "AVERAGE7.VD". [A packed video
C      array contains a 256x256 4-bit/pixel picture in a 64 block file
C      on disk, where 1 block=256 16-bit words (i.e. 1 block=4 video
C      rows). Each 16-bit word holds 4 pixels. Video files are
C      stored  on disk in packed form to conserve disk space. File
C      size is 32K bytes.] Total run time is less than one
C      minute clock time.
C
C      Execution Line Format:          (run SLOWAVE7 on the NOVA)
C            QUICKAVE7
C
C      Loader Command Line Format:
C         RLDR QUICKAVE7 TIMER UNPACK REPACK 20/C @FLIB@
C            The 20/C opens up enough channels to run the
C      program [by default, only 8 channels (designated and pre-
C      designated) are normally available]. Links to A0-A6 in
C      DP0F should be created.
C
C
C******** ARRAY MANAGEMENT **********************************************
C
C      Arrays A0-A6 hold 2 blocks each (i.e. 8 packed video rows)
C
C      Arrays A0U-A6U hold 8 blocks each (i.e. 8 unpacked video rows)
C
C      Array AVE is used for the unpacked averaged picture (8 rows)
C
C      Array AVEP holds the packed averaged picture (8 rows)
C
C      [If array sizes are modified, note that arrays A*U must be dimen-
C      sioned to be 4 times larger than arrays A*.]
C
C            The EQUIVALENCE statement is used to reduce memory
C      requirements. Once an array is unpacked, the packed array is
C      no longer used. Therefore unneeded packed arrays can be used
C      to hold unpacked arrays (but not at the same time!).
C
      INTEGER FILE(2)
      INTEGER A0(512),A1(512),A2(512),A3(512),A4(512),A5(512),
    $  A6(512),AVE(2048),A,B,C,D
      INTEGER A0U(2048),A1U(2048),A2U(2048),A3U(2048),A4U(2048),
    $A5U(2048),A6U(2048),AVEP(512)
      COMMON A0U,A1U,A2U,A3U,A4U,A5U,A6U,AVEP  ;COMMON must be declare
                                               ;before EQUIVALENCE
                                               ;can be used(Fortran IV)
      EQUIVALENCE(AVE,A0U),(A0,A1U),(A1,A2U),(A2,A3U),(A3,A4U),(A4,A5U)
```

```
     $(A5,A6U),(A6,AVEP)
C
C
        TYPE" Program QUICKAVE7 now executing . . .<15><7>"
        CALL TIMER(0)            ;start timer
C
C
C******** I/O FILE MANAGEMENT *******************************************
C
C               Seven channels to "A0" thru "A6" must be OPENed for
C       reading.  An eighth channel to "AVERAGE7.VD" must be OPENed for
C       writing (after "AVERAGE7.VD" is created).
C
        FILE(2)=0
        DO 998 I=0,6
                FILE(1)=16688+I          ;ASCII for A_
                CALL OPEN(I,FILE,1,IER)
                IF(IER.NE.1)WRITE(10,999)FILE(1),IER
 999            FORMAT(" OPEN ",S4," error #",I5)
 998    CONTINUE
        DELETE "AVERAGE7.VD"
        CALL CFILW("AVERAGE7.VD",3,64,IER)       ;create a contiguous
        IF(IER.EQ.1)GO TO 997                    ;file, if possible
        CALL CFILW("AVERAGE7.VD",2,IER) ;create a random file
                                        ;of variable size
        IF(IER.NE.1)TYPE " AVERAGE7.VD create error:",IER STOP
 997    CALL OPEN(7,"AVERAGE7.VD",3,IER)         ;OPEN a channel to
        IF(IER.NE.1)TYPE " OPEN 7 error #",IER   ;"AVERAGE7.VD" for writing
C
C
C
      DO 4 I=0,62,2              ;average 64 blocks, 2 at a time
C***********************************************************************
C
C       Read 2 blocks from each picture to be averaged (8 video rows)
C
        CALL RDBLK(0,I,A0,2,IER)
        CALL RDBLK(1,I,A1,2,JER)
        CALL RDBLK(2,I,A2,2,KER)
        CALL RDBLK(3,I,A3,2,LER)
        CALL RDBLK(4,I,A4,2,MER)
        CALL RDBLK(5,I,A5,2,NER)
        CALL RDBLK(6,I,A6,2,I6ER)
        IF(IER.NE.1..OR.JER.NE.1..OR.KER.NE.1..OR.LER.NE.1..OR.
     $  MER.NE.1..OR.NER.NE.1..OR.I6ER.NE.1)GO TO 5
C
C       Unpack the arrays
C
        CALL UNPACK(512,A0,A0U)
        CALL UNPACK(512,A1,A1U)
        CALL UNPACK(512,A2,A2U)
        CALL UNPACK(512,A3,A3U)
        CALL UNPACK(512,A4,A4U)
        CALL UNPACK(512,A5,A5U)
      CALL UNPACK(512,A6,A6U)
C
C     Perform pixel by pixel averaging
C
      DO 1 K=1,2048
          AVE(K)=IFIX(FLOAT(A0U(K)+A1U(K)+A2U(K)+A3U(K)+A4U(K)+A5U
  $+A6U(K))/7.0+0.5)         ;round to nearest integer
 1    CONTINUE
C
C     Pack array AVE into AVEP and write it to the disk
C
      CALL REPACK(512,AVE,AVEP)
      CALL WRBLK(7,I,AVEP,2,IER)
      IF(IER.NE.1)TYPE " WRBLK #",I," error: ",IER
X     TYPE" Block #",I," averaged.<15>"      ;can be deleted to
                                             ;decrease run time
 4   CONTINUE
C
C     Print total run time message to the console CRT,
C     and EXIT the program.
C
      CALL TIMER(1)          ;stop timer
      GO TO 6
 5    TYPE" <7><15>RDBLK A0 error:",IER,"<15>RDBLK A1 error:",JER,
  $"<15>RDBLK A2 error:",KER,"<15>RDBLK A3 error:",LER,
  $"<15>RDBLK A4 error:",MER,"<15>RDBLK A5 error:",NER,
  $"<15>RDBLK A6 error:",I6ER,"<7><7><15>"
 6    CALL RESET
      TYPE" <15> Program QUICKAVE7 finished."
      TYPE" The averaged file is named AVFRAGE7.VD<15><7><7>"
      STOP
      END
C
C************* Program QUICKAVE7 **************************************
```

```
C***************************************************************************
C
C        Program PICTURE     Written by Lt. Simmons        14 Oct 1981
C        Fortran 5           Revised by Lt. Cromer        12 July 1982
C
C        This program will convert video pixels to lineprinter pixels,
C        and will put the picture in a file or to the Printronix 300
C        lineprinter.  This program prints either the complete 256 by
C        256 pixel picture, or a smaller picture that does not contain
C        the noise created by the video digitizer (the last five blocks
C        in the video file are noise).  Odd numbered Video lines (rows)
C        are represented by 3x3 pixels, while even numbered Video lines
C        are represented by 4x3 pixels.  Run time for a shortened
C        picture of length 222 lines is about 1.5 minutes; the file
C        size will be about 204Kbytes (or 102,000 words). For a 256x256
C        picture, run time should be 2 minutes, with 239Kbytes used on
C        disk.
C
C****************************************************************************
        INTEGER ATITLE(40)
        DIMENSION I1ARRAY(268),I2ARRAY(264),I3ARRAY(201),I4ARRAY(198)
        EQUIVALENCE (I1ARRAY,I2ARRAY,I3ARRAY,I4ARRAY)
        DIMENSION ILP(4,67),IFILE(7),IREC(64),ISAV(4)
        LOGICAL SHORT,TITLE
C
C        Set up solid line, space, and line feed/plot-on characters
C
        IL=177777K                                          ;Solid line
        NC=40100K                                           ;Space
        LF=012K                                             ;Line feed
        LFPC=2412K                                          ;Line feed/plot on
C
C        Set up parameters for complete picture display.
C
        SHORT=.FALSE.                              ;Short picture test
        N1=66                                      ;Top and bottom border length
        N2=256                                     ;Number of lines displayed
        N3=1                                       ;Location of left border
        N4=66                                      ;Location of right border
        N5=67                                      ;Location of line feed
        N6=1                                       ;Length of lines displayed
C
C        Open the video file for input
C
        ACCEPT" What is the name of the input file? "
        READ(11,17)IFILE(1)                                 ;Read video file name
        CALL OPEN(1,IFILE,1,IER)                            ;Open the video file
        CALL CHECK(IER)
        IF(IER.NE.1)TYPE" Input open error:",IER
C
C        Ask for an output file, either the lineprinter or a disk file,
C        and open the disk file if necessary.
C
        ACCEPT" Do you want a disk file created (Y or N)? "
```

```
  1   READ(11,19)N                    ;Read one ASCII character
      IF(N.EQ.19968)GO TO 2           ;File was not selected
      IF(N.NE.22784)GO TO 20          ;Input error
      ACCEPT" A file was selected; what should its name be
     * (13 char max)? "
      READ(11,17)IFILE(1)             ;Read output file name
      CALL DFILW(IFILE,JER)                ;Make sure that the
      IF(JER.NE.1.AND.JER.NE.13)TYPE" Output delete error:",JER
      CALL CFILW(IFILE,2,KER)              ;file does not exist
      IF(KER.NE.1)TYPE" Output create error:",KER
      CALL OPEN(12,IFILE,3,LER)       ;Open the output file
      CALL CHECK(LER)
      IF(LER.NE.1)TYPE" Output file open error:",LER
      GO TO 3
  2   TYPE" The picture will only go to the lineprinter."
C
C     Choose between complete picture and noiseless picture.
C
  3   CONTINUE
      ACCEPT"Do you want to title the picture (Y or N)? "
      READ(11,19)N
      IF(N.EQ.22784)TITLE=.TRUE.
      DO 300 KT=1,40
              ATITLE(KT)=0
 300  CONTINUE
      IF(TITLE)TYPE"Enter title below (up to 80 characters)."
      IF(TITLE)READ(11,3000)ATITLE(1)
 3000 FORMAT(S80)
      ACCEPT"Do you want a complete picture (Y or N)? "
  4   READ(11,19)N               ;Read one ASCII character
      IF(N.EQ.19968)GO TO 22          ;Response was "NO"
      IF(N.NE.22784)GO TO 21          ;Input error
      TYPE" A complete 256 by 256 pixel picture was chosen."
C
C     Put a border at the top of the picture.
C
 5    DO 7 I=1,3
        IF(SHORT)WRITE BINARY(12)NC           ;Space right
        IF(SHORT)WRITE BINARY(12)NC           ;Space right again
        DO 6 J=1,N1
  6     WRITE BINARY(12)IL                    ;Print a line
  7     WRITE BINARY(12)LFPC                  ;Terminate the line
C
C**********************************************************************
C
C     Each line of the picture will have a border on each end.  A
C     DO-LOOP loops 233 (or 256 for whole frame) times around the
C     next three program parts
C
C
      JTEST = -1
      DO 13 JA=1,N2
      JTEST = JTEST * -1
      JJ = 4
```

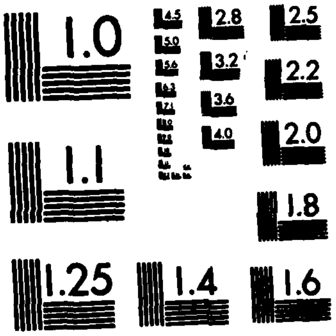
AD-A124 788 SCENE ANALYSIS: NON-LINEAR SPATIAL FILTERING FOR AUTOMATIC TARGET DETECTION(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. J H CROMER 2/2

UNCLASSIFIED DEC 82 AFIT/GE/EE/82D-26 F/G 17/8 NL

END

FILMED

DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
        IF(JTEST.GT.0)JJ=3                         ;Odd iteration
C
C       Put a border down the left hand side
C
        DO 8 K=1,JJ                                ;Put a left border
           IF(SHORT)ILP(K,1)=NC                    ;two spaces in
           IF(SHORT)ILP(K,2)=NC                    ;(~1/4" space)
   8       ILP(K,N3)=43500K                        ;for short picture
C
C       Put a border down the right hand side
C
        DO 9 L=1,JJ                                ;Insert border and
           ILP(L,N4)=40170K                        ;line feed after
   9       ILP(L,N5)=LFPC                                  ;the picture
C
C       Convert the video picture pixels to $LPT pixels
C
           READ BINARY(1)IREC          ;Read one video line
           DO 12 N=N6,64
                IWR=BYTE(IREC(N),2)    ;Right two pixels
                IWL=BYTE(IREC(N),1)    ;Left two pixels
                 IF(JJ.GT.3.5)CALL OUT4X3(IWL,ISAV)
                 IF(JJ.LT.3.5)CALL OUT3X3(IWL,ISAV)
              N7=N+1
              IF(SHORT)N7=N
              DO 11 JB=1,JJ
 11             BYTE(ILP(JB,N7),1)=ISAV(JB)     ;Move to high byte
              IF(JJ.LT.3.5)CALL OUT3X3(IWR,ISAV)
              IF(JJ.GT.3.5)CALL OUT4X3(IWR,ISAV)
              DO 12 JC=1,JJ
 12             BYTE(ILP(JC,N7),2)=ISAV(JC)     ;Store low byte
                L=0
              DO 130 JE=1,JJ
                 DO 130 JD=1,N5
                L=L+1
                I1ARRAY(L)=ILP(JE,JD)
 130            CONTINUE
        IF(N5.EQ.66..AND.JJ.EQ.3)WRITE BINARY(12)I4ARRAY
        IF(N5.EQ.67..AND.JJ.EQ.3)WRITE BINARY(12)I3ARRAY
        IF(N5.EQ.66..AND.JJ.EQ.4)WRITE BINARY(12)I2ARRAY
        IF(N5.EQ.67..AND.JJ.EQ.4)WRITE BINARY(12)I1ARRAY
  13    CONTINUE
C
C*********************************************************************
C
C       Put a border and title at the bottom of the picture
C
        DO 15 JF=1,3
           IF(SHORT)WRITE BINARY(12)NC             ;Space right twice
           IF(SHORT)WRITE BINARY(12)NC             ;for short picture
           DO 14 JG=1,N1
  14       WRITE BINARY(12)IL                      ;Print a line
  15       WRITE BINARY(12)LFPC                    ;Terminate the line
        WRITE BINARY(12)LF                      ;End with a line feed
        WRITE(12,16)ATITLE(1)    ;Title picture
 16     FORMAT(15X,S80)
        CALL RESET                                ;Close all channels
        STOP
C
C       Format statements.
C
C  16   FORMAT(' ',15X,'Signal Processing Laboratory, Air Force
C     1 Institute of Technology, Wright-Patterson AFB, OH  45433<14>')
  17    FORMAT(S13)                               ;Filename format
  18    FORMAT('0')                       ;Double space for short picture
  19    FORMAT(S1)               ;Query format
  20    ACCEPT" Input error.  Try again (YES/NO) > "
        GO TO 1
  21    ACCEPT" Input error.  Try again (YES/NO) > "
        GO TO 4
C
C       Set up parameters for shortened picture.
C
  22    TYPE" The shortened (noise removed) picture was chosen."
 220    ACCEPT"Enter starting row (1-256): ",JSTART
        IF(JSTART.LT.1.OR.JSTART.GT.256)GO TO 220
        ACCEPT" Number of lines to be displayed (255 Max)? ",N2
        N2=MIN(N2,(257-JSTART))
        SHORT=.TRUE.                      ;Turn on short test
        N1=63                             ;Top and bottom border length
        IF(N2.GT.232)N2=228               ;Number of lines displayed
        N3=3                              ;Location of left border
        N4=65                             ;Location of right border
        N5=66                             ;Location of line feed
        N6=4                              ;Length of lines displayed
        IF(JSTART.EQ.1)GO TO 5
        DO 23 J = 1,(JSTART-1)            ;Skip first (JSTART-1)
                                          ;Video lines
  23       READ BINARY(1)IREC
        GO TO 5
        END
C
C********* Program PICTURE *********************************************
```

```
      SUBROUTINE OUT3X3(VIDPIX,LINEPRINT)
C*************************************************************************
C
C     Written by Lt. J.H. Cromer
C
C     This subroutine converts the video pixel values
C     (i.e. an integer value from 0-15) to lineprinter
C     dot matrix form. A 3x3 array pattern
C     is formed by this subroutine. Dot pattern texture
C     (distribution of dots) and average brightness are
C     varied to create 16 pseudo-gray levels. Odd numbered
C     rows of the picture created by PICTURE use
C     these 3x3 patterns.
C     NOTE: The six least significant bits of each byte
C               sent to the P-300 represent print hammer
C               switches (i.e. a 1 turns the hammer on
C               to print a dot, a 0 leaves it off). Bit
C               seven must have a value of 1.
C     (See the Printronix manual for further discussion)
C
C************************************************************************
      INTEGER VIDPIX,LINEPRINT(3),RIGHT,PATTERN(3,16,2)
C
C     Note that right and left pixel patterns are
C     not necessarily the same.
C
      DATA PATTERN/4*7,5,7,6,7,3,7,5,2*2,7,2,5,2,5,2,7,2*2,
     $5,2,5,0,5,2,5,2,5,0,2*2,0,2*2,0,2,1,3*0,2,4*0,
     $4*170K,150K,170K,160K,170K,130K,120K,150K,170K,4*150K,
     $120K,150K,120K,170K,2*120K,150K,120K,150K,100K,150K,
     $140K,120K,110K,120K,100K,150K,110K,140K,110K,100K,150K,
     $2*100K,120K,7*100K/
      RIGHT=(VIDPIX.AND.15)+1
      LEFT=(ISHFT(VIDPIX,-4).AND.15)+1
      DO 10 I=1,3
            LINEPRINT(I)=PATTERN(I,LEFT,1)+PATTERN(I,RIGHT,2)
 10   CONTINUE
      RETURN
      END
C
C********* Subroutine OUT3X3 ***************************************
```

```
      SUBROUTINE OUT4X3(VIDPIX,LINEPRINT)
C***********************************************************************
C
C     Written by Lt. Cromer
C
C     This subroutine returns lineprinter pixels to the
C     calling program PICTURE, which sends video
C     pixels (an integer from 0-15).  The pixel pattern
C     returned is a 4x3 dot matrix array, to be used for
C     the even rows of pictures created by PICTURE.
C     (See OUT3X3.FR for more explanation).
C
C*********************************************************************
      INTEGER VIDPIX,LINEPRINT(4),RIGHT,PATTERN(4,16,2)
      DATA PATTERN/5*7,5,7,7,6,7,7,3,3,6,7,5,5,6,3,2,5,2,
     $3*5,2,2,5,5,0,2,5,2,2,5,2,2,1,4,2,2,4,1,2,1,4,2,0,0,1,
     $4,0,0,2,10*0,5*170K,150K,2*170K,160K,2*170K,2*130K,
     $160K,170K,150K,120K,2*170K,120K,150K,120K,3*150K,2*120K,
     $150K,120K,2*150K,3*120K,150K,2*120K,110K,140K,120K,150K,
     $2*100K,150K,120K,100K,110K,140K,120K,2*100K,
     $120K,2*100K,140K,3*100K,120K,5*100K/
      RIGHT=(VIDPIX.AND.15)+1
      LEFT=(ISHFT(VIDPIX,-4).AND.15)+1
      DO 10 I=1,4
            LINEPRINT(I)=PATTERN(I,LEFT,1)+PATTERN(I,RIGHT,2)
 10   CONTINUE
      RETURN
      END
C
C********* Subroutine OUT4X3 **************************************
```

APPENDIX D:

SCENE AND TEMPLATE SYNTHESIS PROGRAMS

This appendix contains the following programs:

1. REMOVE (UNPACK2, TEST3)
2. EVIDHIST (HISTPLOT)
3. NMOVE (TEST, BLOCK, CHANGE)
4. NEGATE
5. REDUCE
6. TONER

```
C**********************************************************************
C
C       Program REMOVE          by Lt Jim Cromer
C       Fortran 5
C.
C       This program utilizes a 3x3 pixel mask function
C       to remove noise from packed video files.  The mask
C       can be changed by modifying the Subroutine TEST3.
C
C       Execution Line Format:
C               REMOVE infile outfile1 outfile2
C               All files are 64 block video files.
C
C       Load Line Format:
C               RLDR REMOVE TEST3 UNPACK2 IOF @FLIB@
C
C*************************************************************************
        INTEGER A(256),B(256),C(256),D(256),VIDEO(256),FILE1(7),
     $FILE2(7),FILE3(7),THRESH(4),COUNT(16,4),OUT2(256),DUM,MAIN(7),
     $OUT1(256)
        COMMON /COM2/ A,B,C,D,VIDEO
        COMMON /COM3/ THRESH,COUNT,OUT1,OUT2
C
C       I/O FILE MANAGEMENT
C
        CALL IOF(3,MAIN,FILE1,FILE2,FILE3,DUM,DUM,DUM,DUM,DUM)
        CALL OPEN(1,FILE1,2,IER)
        DELETE FILE2
        DELETE FILE3
        CALL CFILW(FILE2,2,IER)
        CALL OPEN(2,FILE2,2,IER)
        CALL CFILW(FILE3,2,IER)
        CALL OPEN(3,FILE3,2,IER)
C
C       Enter 4 threshold values
C
        DO 101 I=1,4
        ACCEPT"ENTER THRESH: ",THRESH(I)
        TYPE"THRESH(",I,")=",THRESH(I)
 101    CONTINUE
C
C       Set top border row values
C
      CALL RDBLK(1,0,VIDEO,1,IER)
      IF(IER.NE.1)STOP " RDBLK #0 error:",IER
       DO 1 I=1,64
               OUT2(I)=VIDEO(I)
 1             OUT1(I)=VIDEO(I)
      CALL UNPACK2(1)
      CALL UNPACK2(2)
C
C       Set pixel value change counters
C
        DO 10 I=1,16
        DO 10 J=1,4
                COUNT(I,J)=0
 10     CONTINUE
C
C       Operate on 64 blocks.  Each call to TEST3
C       operates on 3 rows to output 1 row.  Four calls to
C       TEST3 are made for each packed block read.
C
      DO 2 I=1,63
      CALL TEST3(A,B,C,2)
      CALL TEST3(B,C,D,3)
      CALL RDBLK(1,I,VIDEO,1,IER)
      IF(IER.NE.1)STOP " RDBLK error #",I," error:",IER
      CALL UNPACK2(1)
      CALL TEST3(C,D,A,4)
      CALL WRBLK(2,I-1,OUT1,1,IER)
      IF(IER.NE.1)STOP " WRBLK #",I-1," error:",IER
        CALL WRBLK(3,I-1,OUT2,1,IER)
      CALL TEST3(D,A,B,1)
      CALL UNPACK2(2)
    2 TYPE" <15>Block #",I," tested . . ."
C
      CALL TEST3(A,B,C,2)
      CALL TEST3(B,C,D,3)
C
C       Set bottom border row values
C
      DO 3 I=193,256
        OUT1(I)=VIDEO(I)
    3      OUT2(I)=VIDEO(I)
      CALL WRBLK(2,63,OUT1,1,IER)
      IF(IER.NE.1)TYPE" WRBLK #63 error:",IER
        CALL WRBLK(3,63,OUT2,1,IER)
C
C       Write out # of pixels changed
C
        DO 20 J=1,3,2
        DO 20 I=0,15
        WRITE(12,1000)I+1,I,COUNT(I,J),I,I+1,COUNT(I+1,J+1)
 1000   FORMAT(" #",I2,"S reduced to",I3,":",I5,"  #",I2,
     $"s increased to ",I2,":",I5)
 20     CONTINUE
      TYPE" <15><7><7>Program REMOVE finished.<7>"
      CALL RESET
      STOP
      END
C
C******** Program REMOVE ***********************************
```

```
      SUBROUTINE TEST3(ROW1,ROW2,ROW3,I)
C**************************************************************
C
C     (Called by REMOVE)      by Lt Jim Cromer
C
C     This subroutine determines the noise removed
C     output value of an input pixel by calculating
C     some function of its 8 nearest neighbors, and
C     comparing it with a user input threshold.  Two
C     methods of noise removal are used;  the output
C     arrays are OUT1 and OUT2.
C
C     Packed video blocks hold 4 rows.  The parameter I
C     determines which part of the block that OUT
C     will be packed into.
C             I=1  -->    pack OUT into elements 1-64
C             I=2  -->                         65-128
C             I=3  -->                        129-192
C             I=4  -->                        192-256
C
C*****************************************************************
      INTEGER ROW1(256),ROW2(256),ROW3(256),TEMP1(256)
      INTEGER TEMP2(256),OUT1(256),OUT2(256),THRESH(4)
      INTEGER COUNT(16,4),COLUMN,SURROUND
      LOGICAL TEST
      COMMON /COM3/ THRESH,COUNT,OUT1,OUT2
C
C     Set border values
C
      TEMP1(1)=ROW2(1)
      TEMP2(1)=ROW2(1)
      TEMP1(256)=ROW2(256)
      TEMP2(256)=ROW2(256)
C
C     Check each of the neighbors for a value of
C     the interior pixel +/-1.
C
      DO 20 COLUMN=2,255
      N=1
      TEMP1(COLUMN)=ROW2(COLUMN)
      TEST=ROW2(COLUMN).EQ.0.OR.ROW2(COLUMN).EQ.15
      IF(TEST)GO TO 20
      I2=ROW2(COLUMN)-3
 1    J=0
      I2=I2+2
      DO 5 M=(COLUMN-1),(COLUMN+1)
              IF(ROW1(M).EQ.I2)J=J+1
              IF(ROW2(M).EQ.I2)J=J+1
              IF(ROW3(M).EQ.I2)J=J+1
 5    CONTINUE
      TEST=J.GE.THRESH(N)
      IF(TEST)TEMP1(COLUMN)=I2
      IF(TEST)COUNT(I2,N)=COUNT(I2,N)+1
      IF(TEST)GO TO 20
      N=N+1
      IF(N.EQ.2)GO TO 1
 20   CONTINUE
C
C     Compare the average value of the surround less
C     the interior value with a threshold.
C
      DO 100 COLUMN=2,255
      TEMP2(COLUMN)=ROW2(COLUMN)
      TEST=ROW2(COLUMN).EQ.0.OR.ROW2(COLUMN).EQ.15
      IF(TEST)GO TO 100
      SURROUND=-1*ROW2(COLUMN)
      DO 50 M=(COLUMN-1),(COLUMN+1)
      SURROUND=SURROUND+ROW1(COLUMN)+ROW2(COLUMN)
     $+ROW3(COLUMN)
 50   CONTINUE
      AVERAGE=FLOAT(SURROUND)/8.0
      DIFF=AVERAGE-FLOAT(ROW2(COLUMN))
      I2=ROW2(COLUMN)+1
      I3=INT(AVERAGE+0.5)
      I4=(ROW2(COLUMN)+I3)/2
      I5=MAX(I4,I2)
      I6=MIN(I4,I2-2)
      TEST=DIFF.GE.THRESH(3)
      IF(TEST)TEMP2(COLUMN)=I5
      IF(TEST)COUNT(I5,3)=COUNT(I5,3)+1
      IF(TEST)GO TO 100
      TEST=(-1*DIFF).GE.THRESH(4)
      IF(TEST)TEMP2(COLUMN)=I6
      IF(TEST)COUNT(I2-2,4)=COUNT(I2-2,4)+1
 100  CONTINUE
C
C     Pack the noise removed pixels to OUT1, OUT2
C
      M=-3
      JLOW=(I-1)*64+1
      JHIGH=JLOW+63
      DO 200 J=JLOW,JHIGH
              M=M+4
              OUT1(J)=ISHFT(TEMP1(M),12)+ISHFT(TEMP1(M+1),8)+
     $         ISHFT(TEMP1(M+2),4)+TEMP1(M+3)
              OUT2(J)=ISHFT(TEMP2(M),12)+ISHFT(TEMP2(M+1),8)+
     $         ISHFT(TEMP2(M+2),4)+TEMP2(M+3)
 200  CONTINUE
      RETURN
      END
C
C******** Subroutine TEST3 ******************************************
```

```
      SUBROUTINE UNPACK2(I)
C****************************************************************
C
C     (Called by REMOVE)               by Lt. Jim Cromer
C
C     If I=1 --->     The first two rows of a packed
C                     block are unpacked into arrays
C                     A and B.
C     Else   --->     The last two rows are unpacked
C                     into arrays C and D.
C
C     (See the unpacking subroutines for
C             further explanation).
C
C***************************************************************
      INTEGER A(256),B(256),C(256),D(256),VIDEO(256)
      COMMON /COM2/A,B,C,D,VIDEO
C
      IF(I.EQ.1)GO TO 3
C
      L=-3
      DO 2 K=129,192
         L=L+4
         C(L)=15.AND.ISHFT(VIDEO(K),-12)
         C(L+1)=15.AND.ISHFT(VIDEO(K),-8)
         C(L+2)=15.AND.ISHFT(VIDEO(K),-4)
         C(L+3)=15.AND.VIDEO(K)
         D(L)=15.AND.ISHFT(VIDEO(K+64),-12)
         D(L+1)=15.AND.ISHFT(VIDEO(K+64),-8)
         D(L+2)=15.AND.ISHFT(VIDEO(K+64),-4)
         D(L+3)=15.AND.VIDEO(K+64)
    2 CONTINUE
      GO TO 6
C
    3 L=-3
      DO 5 K=1,64
         L=L+4
         A(L)=15.AND.ISHFT(VIDEO(K),-12)
         A(L+1)=15.AND.ISHFT(VIDEO(K),-8)
         A(L+2)=15.AND.ISHFT(VIDEO(K),-4)
         A(L+3)=15.AND.VIDEO(K)
         B(L)=15.AND.ISHFT(VIDEO(K+64),-12)
         B(L+1)=15.AND.ISHFT(VIDEO(K+64),-8)
         B(L+2)=15.AND.ISHFT(VIDEO(K+64),-4)
         B(L+3)=15.AND.VIDEO(K+64)
    5 CONTINUE
    6 RETURN
      END
C
C********** Subroutine UNPACK2 ***********************************
```

```
C**********************************************************************
C
C       Program EVIDHIST                Written by Lt. Jim Cromer
C       Fortran 5
C.
C       This program calculates a histogram of the 16 gray levels
C       of the input video picture, and types the results on the
C       CRT and/or lineprinter in tabular form.  A plot can be sent
C       to the lineprinter if requested.
C
C       Execution Line Format:          (run VIDHIST on the NOVA)
C               EVIDHIST
C
C       Load Line Format:
C               RLDR EVIDHIST UNPACK HISTPLOT PLOT5.LB @FLIB@
C         A link to PLOT5.LB in DP5F:CALCOM5 should exist or
C         be created before loading.
C
C**********************************************************************
        REAL VALUE(0:15)
        INTEGER INFILE(7),IARRAY(2048),IUNPACK(8192)
C
C******** I/O FILE MANAGEMENT ******************************************
C
 1      ACCEPT"Enter name of video file to be evaluated
     $  --->  "
        READ(11,1000)INFILE(1)
 1000   FORMAT(S13)
        CALL OPEN(1,INFILE,1,IER)
        IF(IER.NE.1)STOP"INFILE OPEN error #",IER
C
C******** INITIALIZE DATA **********************************************
C
        DO 10 I=0,15
                VALUE(I)=0.0
 10     CONTINUE
C
C******** PROCESS THE PICTURE ******************************************
C
        KK=8                    ;This variable determines the # of blocks
                                ;read at a time. (Maximum=16, if array
                                ;sizes are increased.)

        LL=64-KK
        DO 100 J=0,LL,KK
                CALL RDBLK(1,J,IARRAY,KK,IER)
                IF(IER.NE.1)TYPE"1RDBLK #",J," error:",IER
                ISIZE=256*KK    ;256 words/block
                CALL UNPACK(ISIZE,IARRAY,IUNPACK)
                IMAX=ISIZE*4    ;4 pixels/word
                DO 50 I=1,IMAX
                        IPIX=IUNPACK(I)
                        VALUE(IPIX)=VALUE(IPIX)+1.0
 50     CONTINUE
```

```
C
C****** Print the histogram *********
C
 100    CONTINUE
        TYPE"<15>"
        TYPE"Where should the histogram table be printed?"
        TYPE"Enter 1 to send to lineprinter, 2 to send to CRT,"
        TYPE"or 3 for both; any other integer to continue."
        ACCEPT"<11><11>Enter integer option ---> ",IOPT
        TYPE"<15>"
        IF(IOPT.LT.1.OR.IOPT.GT.3)GO TO 160
        IF(IOPT.EQ.1.OR.IOPT.EQ.3)ICH=12
 110    IF(IOPT.EQ.2)ICH=10
                IF(ICH.EQ.10)WRITE(10,5000)INFILE(1)
                IF(ICH.EQ.12)WRITE(ICH,2000)INFILE(1)
                DO 120 J=0,7
                JJ=J+8
                WRITE(ICH,3000)J,VALUE(J),JJ,VALUE(JJ)
 120            CONTINUE
        IF(ICH.EQ.12)WRITE(12,4000)
        IF(IOPT.EQ.3)IOPT=2
        IF(IOPT.EQ.2.AND.ICH.EQ.12)GO TO 110
 160    TYPE"<15>"
C
C****** Plot the histogram **********
C
        TYPE"Enter 1 to plot histogram on lineprinter, any "
        ACCEPT"other integer to continue: ",I
        IF(I.NE.1)GO TO 200
        CALL HISTPLOT(VALUE,INFILE)
C
C***** EXIT program or start over *********
C
 200    CALL RESET
 2000   FORMAT(///////27X," <10>HISTOGRAM OF ",S13,/25X,
     $" ——————————————————————————————"//)
 3000   FORMAT(18X," # of level",I2,":",I5,5X," # of level",I3,":",I5)
 4000   FORMAT("0")
 5000   FORMAT(//27X,"HISTOGRAM OF ",S13,/)
        TYPE"<15>"
        ACCEPT"Enter 1 to evaluate another file, any other integer
     $ to stop:  ",I
        IF(I.EQ.1)GO TO 1
        STOP
        END
C
C************ Program EVIDHIST ****************************************
```

```
      SUBROUTINE HISTPLOT(VALUE,INFILE)
C***********************************************************************
C
C     Written by Lt. Jim Cromer
C     This subroutine plots the 16 element array VALUE
C     passed to it as a histogram.  It is called by the
C     program EVIDHIST.
C
C***********************************************************************
      REAL VALUE(0:15),INFILE(7)
      REAL X(481),Y(481)
      Y(1)=0.0
      K=1
      X(1)=-0.25              ;X-axis starting point
C
C******** Create the arrays to be plotted ***************************
C
      VNORM=655.36            ;# pixels/100
 1    ACCEPT"Enter histogram level to be suppressed
     $ (999 to continue): ",ISUPPRESS
      IF(ISUPPRESS.EQ.999)GO TO 3
      IF(ISUPPRESS.LT.0.OR.ISUPPRESS.GT.15)TYPE"INPUT ERROR!<7><15>"
      IF(ISUPPRESS.LT.0.OR.ISUPPRESS.GT.15)GO TO 1
      VNORM=VNORM-VALUE(ISUPPRESS)/100.0
      VALUE(ISUPPRESS)=0.0
      GO TO 1
  3   DO 20 I=0,15            ;do 16 values
            DO 5 J=1,15       ;determines width of bars
            K=K+1
            X(K)=X(K-1)+1.0/30.0
            Y(K)=VALUE(I)/VNORM
  5         CONTINUE
            X(K-15)=X(K-14)
            DO 10 J=16,30     ;determines spacing between bars
            K=K+1
            X(K)=X(K-1)+1.0/30.0
            Y(K)=0.0
 10         CONTINUE
 20   CONTINUE
C
C     Find end of input filename, then insert blanks after it
C
      DO 30 J=2,13
            IF((BYTE(INFILE,J)).EQ.0)BYTE(INFILE,J)=32
            IF((BYTE(INFILE,(J-1))).EQ.32)BYTE(INFILE,J)=32
 30   CONTINUE
      Y(480)=0.0
      X(480)=16.0
      XAX=5.0           ;set X- and Y-axes length
      YAX=4.0
C
C******** Plot the arrays *******************************************
C
      TYPE"Please wait while a plot is generated (50 secs)."
```

```
          CALL PLOTS(0,0,6)
          CALL PLOT(1.5,5.0,-3)
          CALL ASCALE(X,XAX,480,1,FX,DX)  ;scale the arrays
          CALL ASCALE(Y,YAX,480,1,FY,DY)
C
C         Title the axes
C
          CALL AXIS(0.0,0.0,"PIXEL VALUE",-11,XAX,0.0,FX,DX)
          CALL AXIS(0.0,0.0,"PERCENTAGE OF EVALUATED
      $ PIXELS",30,YAX,90.0,FY,DY)       ;vertical axis title
          CALL ALINE(X,Y,480,1,0,1,FX,DX,FY,DY)
C
C         Title the plot
C
          CALL SYMBOL(0.1,4.0,0.2,"HISTOGRAM OF ",0.0,13)
          CALL SYMBOL(3.1,4.0,0.2,INFILE,0.0,13)
          CALL PLOT(0.0,0.0,999)  ;send to lineprinter
          RETURN
          END
C
C************ Subroutine HISTPLOT ***********************************
```

```
C****************************************************************
C
C       Program TONER           Written by Lt. Jim Cromer
C
C       This program will convert individual pixel values
C       of an input video file into new values assigned by
C       the user.  It can be used to adjust gray-levels
C       (i.e. histogram equalization), increase contrast,
C       display selected pixel values, create "negative im-
C       ages", create files of a constant gray-level, and for
C       many other purposes. (NOTE: The program EVIDHIST can
C       be run to generate a histogram of any VIDEO file.)
C
C       Execution Line Format:
C               TONER
C               The program will ask for the input and output
C         file names, and the new pixel values.
C
C       Load Line Format:
C               RLDR TONER TIMER UNPACK REPACK @FLIB@
C
C***************************************************************
        INTEGER NEWVALUE(0:15),INFILE(7),OUTFILE(7)
        INTEGER PACKED(4096),UNPACKED(16384)
        ISTART=0                ;start timer
        ISTOP=1                 ;stop timer

C
C******** USER INPUT OF VARIABLES ******************************
C
        ACCEPT"Enter name of video file to be toned ---> "
        READ(11,1000)INFILE(1)
        ACCEPT"Enter name of output file ---> "
        READ(11,1000)OUTFILE(1)
        TYPE"<15>","Enter new pixel values<15>"
        TYPE"   NOTE: Leading zeros are significant, i.e. enter"
        TYPE"   a '1' as '01', enter a '2' as '02', etc.<15>"
        DO 1 J=0,15             ;16 gray-levels
                WRITE(10,2000)J
                READ(11,3000)NEWVALUE(J)
                IF(NEWVALUE(J).LT.0.OR.NEWVALUE(J).GT.15)
     $NEWVALUE(J)=15
 1      CONTINUE
        CALL TIMER(ISTART)
C
C******** I/O FILE MANAGEMENT *********************************
C
        CALL OPEN(1,INFILE,1,IER)
        IF(IER.NE.1)TYPE"INFILE OPEN error #",IER
        CALL DFILW(OUTFILE,IER)
        IF(IER.NE.1.AND.IER.NE.13)TYPE"OUTFILE DFILW
     $ error #",IER
        CALL CFILW(OUTFILE,3,64,IER)    ;create a contiguous file
        IF(IER.EQ.41)CALL CFILW(OUTFILE,2,IER)
        IF(IER.NE.1)TYPE"OUTFILE CFILW error #",IER
        CALL OPEN(2,OUTFILE,3,IER)
        IF(IER.NE.1)TYPE"OUTFILE OPEN error #",IER
C
C******** RE-TONE THE PICTURE ********************************
C
        DO 3 J=0,48,16
                CALL RDBLK(1,J,PACKED,16,IER)
                IF(IER.NE.1)TYPE"RDBLK #",J," error:",IER
                CALL UNPACK(4096,PACKED,UNPACKED)
                DO 2 I=1,16384  ;do 1/4 of picture
                        UNPACKED(I)=NEWVALUE(UNPACKED(I))
 2              CONTINUE
                CALL REPACK(4096,UNPACKED,PACKED)
                CALL WRBLK(2,J,PACKED,16,IER)
                IF(IER.NE.1)TYPE"WRBLK #",J," error:",IER
 3      CONTINUE
C
C       Send message to CRT terminal
C
        CALL TIMER(ISTOP)
        WRITE(10,4000)OUTFILE(1)
        TYPE"<15>","Have a nice day!<7><15>"
C
C********* WRITE NEW TONE VALUES TO THE LINEPRINTER ***************
C
        WRITE(12,5000)
        WRITE(12,6000)INFILE(1),OUTFILE(1)
        DO 5 I=0,15
                WRITE(12,7000)I,NEWVALUE(I)
 5      CONTINUE
 1000   FORMAT(S13)
 2000   FORMAT(" Change old pixel value",I3," to ?")
 3000   FORMAT(I2)
 4000   FORMAT(" The toned picture is in the file ——>  ",S13)
 5000   FORMAT(//////26X," RESULTS OF TONER<10>"/
     $26X," ————————————————"////)
 6000   FORMAT(10X," Input file ——> ",S13,/10X," Output file
     $——> ",S13,//20X,"OLD PIXEL",10X,"NEW PIXEL",/20X,
     $"—————————",10X,"—————————"/)
 7000   FORMAT(23X,I2,17X,I2)
        CALL RESET
        STOP
        END
C
C*********** Program TONER ******************************************
```

```
C***************************************************************************
C
C       Program NMOVE           Written by Lt. Jim Cromer
C       Fortran IV              16 Aug 1982
C.
C       This program will place the video information in the win-
C       dow given for the template (inputfile1) inside of the
C       window given for the background (inputfile2), and write
C       the combined picture to the outputfile.  The window may be
C       placed anywhere within the background, and may be taken
C       from anywhere within the template.  Window width, length,
C       and position are input by the user.
C
C       Execution line format:  (on the NOVA only)
C               NMOVE           (run EMOVE on the ECLIPSE)
C
C       Loader command line format (NOVA only):
C       RLDR NMOVE TEST BLOCK CHANGE XWRBLK XRDBLK
C               UNPACK REPACK FORT.LB
C
C***************************************************************************
C
        INTEGER IPAR(2),INFILE1(7),INFILE2(7),OUTFILE(7),
     $CB1,CBLOCKS,CCOL,CLS,CSTOP,CTOP,CLB,CLEFT,TTOP,TB1,TBLOCKS,CH3,
     $TCOL,TLS,TSTOP,TLB,TLEFT,COMB(1024),TEMP(1024),BACK(1024),WIDTH,TB
        COMMON/LIST1/COMB,TEMP,CLS,TLS
        COMMON/LIST2/LENGTH,WIDTH
        EQUIVALENCE (COMB,BACK)
C
C
C******* I/O FILE MANAGEMENT *************************************
C
        TYPE"<15>","Program NMOVE is to be run on the NOVA only!"
 99     TYPE"<15>","*********************************************<15>"
        ACCEPT" Enter template file name: "
        READ(11,1000)INFILE1(1)
        ACCEPT"<15>"," Enter background file name: "
        READ(11,1000)INFILE2(1)
        DO 999 J=1,7
 999            OUTFILE(J)=INFILE2(J)
        ACCEPT"<15>"," Enter combined output file name: "
        READ(11,1000)OUTFILE(1)
 1000   FORMAT(S13)
        CALL OPEN(1,INFILE1,2,IER)
        IF(IER.NE.1)TYPE" Channel 1 OPEN error:",IER
        CALL OPEN(2,INFILE2,2,IER)
        IF(IER.NE.1)TYPE" Channel 2 OPEN error:",IER
        CH3=2
        ICOUNT=0
        DO 1002 J=1,7           ;check for BACKGROUND=COMBINED
 1002           IF(OUTFILE(J).EQ.INFILE2(J))ICOUNT=ICOUNT+1
        IF(ICOUNT.EQ.7)GO TO 1
        CH3=3
        CALL DFILW(OUTFILE,IER)
```

```
      IF(IER.NE.1.AND.IER.NE.13)TYPE"OUTFILE DFILW error:",IER
      CALL CFILW(OUTFILE,2,IER)
      IF(IER.NE.1)TYPE" CFILW error:",IER
      CALL OPEN(CH3,OUTFILE,2,IER)
      IF(IER.NE.1)TYPE" Channel 3 OPEN error:",IER
C
C
C******* ENTER WINDOW PARAMETERS ******************************
C
 1    ACCEPT"<15>"," Enter top row of template window (1-256):",TTOP
      ACCEPT" Enter left column of template window (1-256):",TLEFT
      ACCEPT" Enter width of window (1-256):",WIDTH
      ACCEPT" Enter length of window (1-256):",LENGTH
C
C     The calls to TEST check to see if the input parameters are
C     legal, and modifies them if necessary:
C             0 < TOP < 257,     (TOP + LENGTH) < 258
C             0 < LEFT < 257,    (LEFT + WIDTH) < 258
C
      CALL TEST(TTOP,TLEFT)
      ACCEPT"<15>"," Enter top row of background window (1-256):",CTOP
      ACCEPT" Enter left column of background window (1-256):",CLEFT
      CALL TEST(CTOP,CLEFT)
      CALL BLOCK(TBLOCKS,TB1,TLS,TCOL,TTOP,TLEFT)
      CALL BLOCK(CBLOCKS,CB1,CLS,CCOL,CTOP,CLEFT)
C
C     Determine column number of the last video row (0-3)
C
      J1=MOD(LENGTH,4)
      TSTOP=MOD((TCOL+J1),4)-1
      CSTOP=MOD((CCOL+J1),4)-1
      IF(CSTOP.EQ.-1)CSTOP=3
      IF(TSTOP.EQ.-1)TSTOP=3
C
C     Determine the last significant block of window
C
      CLB=CB1+CBLOCKS-1
      TLB=TB1+TBLOCKS-1
C
C     User check of window parameters
C
      TYPE"<15>","WIDTH=",WIDTH,"     LENGTH=",LENGTH
      TYPE"TEMPLATE TOP ROW=",TTOP,"  BACKGROUND TOP ROW=",CTOP
      TYPE"TEMPLATE LEFT COLUMN=",TLS,"  BACKGROUND LEFT COLUMN
    $=",CLS,"<15>"
      ACCEPT"Enter 1 to see expanded set of variables, any
    $ other integer to continue:  ",I
      IF(I.NE.1)GO TO 5
      TYPE"**************************************************"
      TYPE" PARAMETER    TEMPLATE  BACKGROUND"
      TYPE" _________    ________  __________"
      TYPE"<15>"," TOP ROW    ",TTOP,CTOP
      TYPE"<15>"," START COL #",TCOL,CCOL
      TYPE"<15>"," STOP COL # ",TSTOP,CSTOP
```

```
        TYPE"<15>"," FIRST BLOCK",TB1,CB1
        TYPE"<15>"," LAST BLOCK ",TLB,CLB
        TYPE"<15>"," # OF BLOCKS",TBLOCKS,CBLOCKS
        TYPE"<15>"," LEFT COL    ",TLS,CLS
        TYPE"<15>"," WIDTH= ",WIDTH
        TYPE"<15>"," LENGTH=",LENGTH
        TYPE"<15>","***************************************************"
        ACCEPT" Enter 1 to try another set, any other integer
     $ to continue: ",I
        IF(I.EQ.1)GO TO 1
C
C
C****** Create the combined picture **************************
C
  5     ICOUNT=0
        IF(CH3.EQ.2)GO TO 20    ;If combined picture file
                                ;is the same as the back-
                                ;ground picture file, then no
                                ;need to write to itself
C       Write background only blocks (before window)
C       to the combined picture file.
C
        JMAX=CB1-1
        IF(JMAX.LT.0)GO TO 20
        DO 10 J=0,JMAX
                CALL RDBLK(2,J,BACK,1,IER)
                IF(IER.NE.1)TYPE" 2RDBLK",J," error:",IER
                CALL WRBLK(CH3,J,COMB,1,IER)
                IF(IER.NE.1)TYPE" WRBLK",J," error:",IER
                ICOUNT=ICOUNT+1
  10    CONTINUE
  20    TYPE" Background before window completed."
        TYPE" # Blocks written:",ICOUNT
C
C
C . . .  Overlay template window onto background  . . . . .
C
        CALL XRDBLK(1,TB1,TEMP,1,IER)
        IF(IER.NE.1)TYPE"1RDBLK #",TB1," error:",IER
        CALL XRDBLK(2,CB1,BACK,1,IER)
        IF(IER.NE.1)TYPE"2RDBLK #",CB1," error:",IER
        N1=TCOL         ;4-MAX(N1,N2) gives the number of rows
        N2=CCOL         ;to change before the next RDBLK
        IF(TCOL.GT.CCOL)GO TO 100
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
C       There are four columns in the packed video array (64x4),
C       designated 0, 1, 2, and 3.  If the template starting
C       column number is less than or equal to the background (combined)
C       starting column number, then the background block will be "used
C       up" before the template block.  When the background block is
C       finished, a WRBLK is done, and the next background block is read
C       When the template block is finished, the next template block is
C       read, but no WRBLK needs to be performed.  Note that the back-
```

```
C       ground and combined files are always at the same block number.
C
        CALL CHANGE(N2,N2,N1)
        CALL XWRBLK(CH3,CB1,COMB,1,IER)
        IF(IER.NE.1)TYPE" WRBLK #",CB1," error:",IER
C
C       Write the template window into the background
C
        TB=TB1+1
        IMIN=CB1+1
        ICOUNT=1
        IMAX=CLB-1
        IF(IMIN.GT.IMAX)GO TO 60
        DO 50 I=IMIN,IMAX
                CALL XRDBLK(2,I,BACK,1,IER)
                IF(IER.NE.1)TYPE" 2RDBLK #",I," error:",IER
                CALL CHANGE(N1,N2,N1)
                CALL XRDBLK(1,TB,TEMP,1,IER)
                IF(IER.NE.1)TYPE" 1RDBLK #",TB," error:",IER
                CALL CHANGE(N2,N2,N1)
                CALL XWRBLK(CH3,I,COMB,1,IER)
                IF(IER.NE.1)TYPE" WRBLK #",I," error:",IER
                ICOUNT=ICOUNT+1
 50     TB=TB+1
 60     TYPE" TOOL.LT.COOL--Window portion complete."
        TYPE" # blocks written:",ICOUNT
        GO TO 250
C
C       In this case the template starting column number
C       is greater than the background starting column number.
C       The template block must be "finished" first.
C
100     CALL CHANGE(N1,N2,N1)           ;finish TEMP block
        TB=TB1+1
        IMAX=CLB-1
        ICOUNT=0
        IF(CB1.GT.IMAX)GO TO 225
        DO 200 I=CB1,IMAX
                CALL XRDBLK(1,TB,TEMP,1,IER)
                IF(IER.NE.1)TYPE" 1RDBLK #",TB," error:",IER
                CALL CHANGE(N2,N2,N1)   ;finish BACK block
                CALL XWRBLK(CH3,I,COMB,1,IER)
                IF(IER.NE.1)TYPE" WRBLK #",I," error:",IER
                ICOUNT=ICOUNT+1
                IBLK=I+1
                CALL XRDBLK(2,IBLK,BACK,1,IER)
                IF(IER.NE.1)TYPE" 2RDBLK #",IBLK," error:",IER
                CALL CHANGE(N1,N2,N1)   ;finish TEMP block
200     TB=TB+1
225     TYPE" TOOL.GT.COOL--Window portion complete."
        TYPE" # blocks written:",ICOUNT
C
C
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
C
C     If the combined (background) stopping column number
C     is greater than the template stop column number, then the
C     second last template block (I2LB=TLB-1) must be read (i.e.
C     there are more video rows to be changed in the last back-
C     ground block than there are available in the last template
C     block to change them to).  If TSTOP is greater than or equal
C     to CSTOP, then there are sufficient video rows available in
C     the last template block to complete the last
C     background block to be changed.
C
 250  IF(CSTOP.GT.TSTOP)GO TO 400
      M1=TSTOP-CSTOP
      CALL XRDBLK(1,TLB,TEMP,1,IER)
      IF(IER.NE.1)TYPE" 1RDBLK #",TLB," error:",IER
      CALL XRDBLK(2,CLB,BACK,1,IER)
      IF(IER.NE.1)TYPE" 2RDBLK #",CLB," error:",IER
      N1=3-CSTOP
      N2=0
      CALL CHANGE(N1,N2,M1)   ;finish BACK block to CSTOP
      CALL XWRBLK(CH3,CLB,COMB,1,IER)
      IF(IER.NE.1)TYPE" WRBLK #",CLB," error:",IER
      TYPE" CSTOP.LT.TSTOP—Last block of window complete."
      GO TO 500
C
C     Complete the last block of the window
C     NOTE:  CSTOP is greater than TSTOP.  Therefore finish
C            TEMP before BACK.
C
 400  M1=CSTOP-TSTOP
      CALL XRDBLK(2,CLB,BACK,1,IER)
      IF(IER.NE.1)TYPE" 2RDBLK #",CLB,"  error:",IER
      I2LB=TLB-1
      CALL XRDBLK(1,I2LB,TEMP,1,IER)
      IF(IER.NE.1)TYPE" 1RDBLK #",I2LB," error:",IER
      N1=4-M1
      N2=0
      CALL CHANGE(N1,N2,N1)          ;finish TEMP block
      CALL XRDBLK(1,TLB,TEMP,1,IER)
      IF(IER.NE.1)TYPE" 1RDBLK #",TLB," error:",IER
      M1=3-TSTOP
      CALL CHANGE(M1,N2,N1)          ;finish BACK block to CSTOP
      CALL XWRBLK(CH3,CLB,COMB,1,IER)
      IF(IER.NE.1)TYPE" WRBLK #",CLB," error:",IER
      TYPE" CSTOP.GT.TSTOP—Last block of window complete."
 500  ICOUNT=1
C
C     Finish the combined file (background only portion)
C
      JMIN=CLB+1
      IF(JMIN.GT.63)GO TO 601        ;if finished STOP
      IF(CH3.EQ.2)GO TO 601          ;if COMBINED=BACKGROUND, STOP
      DO 600 J=JMIN,63
              CALL RDBLK(2,J,BACK,1,IER)
```

```
                IF(IER.NE.1)TYPE" 2RDBLK #",J," error:",IER
                CALL WRBLK(CH3,J,COMB,1,IER)
                IF(IER.NE.1)TYPE" WRBLK #",J," error:",IER
                ICOUNT=ICOUNT+1
 600    CONTINUE
        TYPE" Finished background only portion."
 601    TYPE" # blocks written:",ICOUNT
        TYPE"<15>","<7>","<15>"," Program NMOVE execution completed. <7>"
        WRITE(10,2000)OUTFILE(1)
 2000   FORMAT(" The combined picture is in the file --> ",S13)
C
C
C****** Present Option Menu ***********************************
C
        GO TO 2010
 2002   TYPE"<15>","Input error.  Try again."
 2010   TYPE"<15>","********************************************"
        TYPE"<15>","What next?<15>","Here are the options:"
        TYPE"<15>","<11>1 - Try another set of window values"
        TYPE"<15>","<11>2 - Start over with new input pictures"
C       TYPE"<15>","<11>3 - Display combined picture on the video monitor"
        TYPE"<15>","<11>3 - Save combined picture and STOP<15>"
        ACCEPT"<11>Enter option ---> ",IOPT
        IF(IOPT.LT.1.OR.IOPT.GT.3)GO TO 2002
        IF(IOPT.EQ.1)GO TO 1
        CALL RESET
        IF(IOPT.EQ.2)GO TO 99
        IF(IOPT.EQ.3)STOP
        TYPE"<15>","Check monitor - - Press green CHOPS control
     $button to continue."
        IDCNT=4
        IPAR(1)=9999
        IPAR(2)=0
        WRITE(10,3000)OUTFILE(1)
 3000   FORMAT("0","Picture being displayed  ---> ",S13)
C       CALL CHANNEL(0,0,3,0,0,"A",0,0,0,IE,IS)         ;call abort
C       CALL CHANNEL(3,1,2,1,IDCNT,OUTFILE,64,0,IPAR,IERR,ISYS)
C       CALL ERCHK(IERR,1,IDCNT,1,ISYS)
        TYPE"<15>","CHANNEL currently not loaded."
        TYPE"Use VIDEO to display combined pictures.<15>"
        CALL OPEN(1,INFILE1,2,IER)                ;re-OPEN channels
        IF(IER.NE.1)TYPE"CH1 RE-OPEN ERROR:",IER
        CALL OPEN(2,INFILE2,2,IER)
        IF(IER.NE.1)TYPE"CH2 RE-OPEN ERROR:",IER
        IF(CH3.EQ.3)CALL OPEN(3,OUTFILE,2,IER)
        IF(IER.NE.1)TYPE"CH3 RE-OPEN ERROR:",IER
        GO TO 2010
        END
C
C********* Program NMOVE *********************************************
```

```
                SUBROUTINE TEST(TOP,LEFT)
C*******************************************************************
C
C       Subroutine TEST checks to see if the input parameters
C       to program NMOVE are legal, and modifies them if
C       necessary.  (It is also called by DISTANCE.)
C
C*******************************************************************
        INTEGER TOP,WIDTH
        COMMON/LIST2/LENGTH,WIDTH
        IF(LEFT.LT.1.OR.LEFT.GT.256)LEFT=1
        MAXWIDTH=257-LEFT               ;picture has 256 columns
        IF(WIDTH.GT.MAXWIDTH.OR.WIDTH.LT.1)WIDTH=MAXWIDTH
        IF(TOP.LT.1.OR.TOP.GT.256)TOP=1
        MAXLENGTH=257-TOP               ;picture has 256 rows
        IF(LENGTH.GT.MAXLENGTH.OR.LENGTH.LT.1)LENGTH=MAXLENGTH
        RETURN
        END
C
C********* Subroutine TEST *****************************************
```

```
        SUBROUTINE BLOCK(NUMBLOCKS,BLOCK1,LEFTSIDE,COLUMN,TOP,LEFT)
C***********************************************************************
C
C       Subroutine BLOCK determines the total number of blocks to
C       be read into the window, the first block to be read,
C       and the first video row "column" number.  This subroutine
C       is called by NMOVE.
C
C***********************************************************************
        INTEGER BLOCK1,COLUMN,TOP,REMAINDER,WIDTH
        COMMON/LIST2/LENGTH,WIDTH
        BLOCK1=INT((TOP-1)/4.0)         ;4 rows per block
        COLUMN=MOD((TOP-1),4)
        LEFTSIDE=LEFT
        REMAINDER=MOD(LENGTH,4)
        K1=LENGTH+3
        NUMBLOCKS=INT(K1/4.0)
        IF(REMAINDER.EQ.2.AND.COLUMN.GT.2)NUMBLOCKS=NUMBLOCKS+1
        IF(REMAINDER.EQ.3.AND.COLUMN.GT.1)NUMBLOCKS=NUMBLOCKS+1
        IF(REMAINDER.EQ.0.AND.COLUMN.GT.0)NUMBLOCKS=NUMBLOCKS+1
        IF(NUMBLOCKS.GT.1)RETURN
        TYPE"WARNING:  # Blocks to be read =",NUMBLOCKS
        PAUSE
        RETURN
        END
C
C************ Subroutine BLOCK *************************************
```

```
      SUBROUTINE CHANGE(JMIN,CSTART,TSTART)
C**************************************************************************
C
C     Written by Lt. Jim Cromer
C     Subroutine CHANGE changes the corresponding background
C     (i.e. the combined picture) pixels to template pixels;
C     it is called by the program NMOVE.
C
C**************************************************************************
      INTEGER COMB(1024),TEMP(1024),CLS,TLS,CSTART,TSTART,WIDTH
      COMMON /LIST2/ LENGTH,WIDTH
      COMMON /LIST1/ COMB,TEMP,CLS,TLS
C
      DO 2 J=JMIN,3
              K=TSTART*256+TLS        ;Set left side of input(template
              M=CSTART*256+CLS        ;and output (combined) windows.
              KMAX=K+WIDTH-1  ;Change values over the width of window
              DO 1 L=K,KMAX
                      COMB(M)=TEMP(L)
  1                   M=M+1
      TSTART=TSTART+1
  2   CSTART=CSTART+1
      IF(CSTART.EQ.4)CSTART=0         ;reset row-pointer if necessary
      IF(TSTART.EQ.4)TSTART=0
      RETURN
      END
C
C********* Subroutine CHANGE *********************************************
```

```
C*********************************************************************
C
C       Program NEGATE                     by Lt. Jim Cromer
C       Fortran 5
C
C       This program writes to an output video file the "negative"
C       pixels values of the input video file (i.e. dark pixels
C       become light, and vice-versa).  It will also "flip" the
C       picture about its horizontal axis (i.e. turn it upside down).
C
C       Execution Line Format:
C               NEGATE[/F] infile outfile
C         The /F is chosen if the picture is to be reversed
C         horizontally.  The program can be run twice to produce
C         a horizontally flipped positive image.
C
C       Load Line Format:
C               RLDR NEGATE IOF UNPACK REPACK TIMER @FLIB@
C
C**********************************************************************
        INTEGER INFILE(7),OUTFILE(7),GS(2),NEW(1024),OLD(1024)
        INTEGER VIDEO(256),DUM,MAIN(7)
C
C******** I/O FILE MANAGEMENT ****************************************
C
C
        CALL IOF(2,MAIN,INFILE,OUTFILE,DUM,GS,DUM,DUM,DUM)
        CALL TIMER(0)           ;start timer
        OPEN 1,INFILE
        DELETE OUTFILE
        CALL CFILW(OUTFILE,3,64,IER)
        IF(IER.EQ.1)TYPE"Contiguous file created"
        IF(IER.EQ.41)CALL CFILW(OUTFILE,2,IER)
        IF(IER.NE.1)STOP " Random file creation error "
        OPEN 2, OUTFILE
C
C******** Test switch and set variables ****************************
C
        IF(GS(1).EQ.1024)GO TO 1        ;test global switch
        TYPE" Creating negative video file only."
        I1=0
        I2=256
        I3=512
        I4=768
        GO TO 2
   1    TYPE" Creating negative and horizontally flipped video file."
        I1=768
        I2=512
        I3=256
        I4=0
C
C******** Loop around the next section 64 times
C         to process the entire picture           *********************
C
```

```
  2     DO 4 I=0,63              ;do entire picture (64 blocks)
        K=I
        IF(I1.EQ.768)K=63-I      ;if flipped, start at bottom of infile
        CALL RDBLK(1,K,VIDEO,1,IER)
        IF(IER.NE.1)TYPE " RDBLK error #:",IER STOP
        CALL UNPACK(256,VIDEO,OLD)
C
C       Negate pixels, and re-arrange if required.
C       Work on 4 rows at a time.
C
                DO 3 N=1,256
                NEW(N)=15-OLD(N+I1)
                NEW(N+256)=15-OLD(N+I2)
                NEW(N+512)=15-OLD(N+I3)
                NEW(N+768)=15-OLD(N+I4)
  3             CONTINUE
        CALL REPACK(256,NEW,VIDEO)
        CALL WRBLK(2,I,VIDEO,1,IER)
        IF(IER.NE.1)STOP" WRBLK error #",IER
  4     CONTINUE
C
C******* Write completion message to CRT ************
C
        CALL RESET
        CALL TIMER(1000)
        WRITE(10,5)OUTFILE(1)
  5     FORMAT(" The negative picture is in the file ---> ",S13)
        STOP
        END
C
C************* Program NEGATE ******************************************
```

```
C**********************************************************************
C
C       Program REDUCE          Written by Lt. Jim Cromer
C       FORTRAN 5
C
C       This program reduces a 256x256 pixel video picture into
C       a 128x128 array by averaging 4 pixels into 1.  The
C       reduced array is placed in the upper left-hand quadrant;
C       all other pixels in the output picture are made white for
C       display purposes.
C
C       Execution Line Format:
C               REDUCE infile outfile
C          Both the input and output files are 256x256 video files
C       (i.e. packed integer form, 64 blocks per file).
C
C       Load Line Format:
C               RLDR REDUCE IOF TIMER UNPACK REPACK @FLIB@
C
C***********************************************************************
        INTEGER OLDPACKED(4096),OLDUNPACK(16384),NEWUNPACK(8192)
        INTEGER NEWPACKED(2048),INFILE(7),OUTFILE(7),MAIN(7)
        COMMON OLDPACKED,OLDUNPACK
        EQUIVALENCE (OLDUNPACK,NEWUNPACK),(NEWPACKED,OLDPACKED)
C
C
C******** I/O FILE MANAGEMENT ****************************************
C
        CALL IOF(2,MAIN,INFILE,OUTFILE,I1,I2,I3,I4,I5)
        CALL TIMER(0)           ;start timer
        CALL DFILW(OUTFILE,IER)
        IF(IER.NE.1.AND.IER.NE.13)TYPE"OUTFILE DFILW error:",IER
        CALL CFILW(OUTFILE,3,64,IER)
        IF(IER.EQ.41)CALL CFILW(OUTFILE,2,IER)
        IF(IER.NE.1)TYPE"OUTFILE CFILW error:",IER
        CALL OPEN(1,INFILE,1,IER)
        IF(IER.NE.1)TYPE"INFILE OPEN error:",IER
        CALL OPEN(2,OUTFILE,3,IER)
        IF(IER.NE.1)TYPE"OUTFILE OPEN error:",IER
C
C******** Process the picture ****************************************
C
        DO 400 M=0,3
                M1=M*16         ;RDBLK counter
                CALL RDBLK(1,M1,OLDPACKED,16,IER);read 64 rows
                IF(IER.NE.1)TYPE"RDBLK #",M1," error:",IER
                CALL UNPACK(4096,OLDPACKED,OLDUNPACK)
                K=0             ;new row counter
C
C       This section reduces 64 rows of 256 elements each into
C       32 rows of 128 elements each.  It executes this function
C       4 times so that a total of 256 rows are reduced to 128.
C
                DO 300 J=1,32           ;create 32 new rows from 64
                DO 100 I=1,128  ;create 128 elements/new row
                        K=K+1
                        L=K*2           ;old row 1 counter
                        LL=L+256        ;old row 2 counter
                        A1=OLDUNPACK(L-1)
                        A2=OLDUNPACK(L)
                        A3=OLDUNPACK(LL-1)
                        A4=OLDUNPACK(LL)        ;average the 4 pixels
                        NEWUNPACK(K)=IFIX((A1+A2+A3+A4)/4.0+0.5)
 100                    CONTINUE
                        IMIN=K+1
                        IMAX=K+128
                DO 200 I=IMIN,IMAX              ;finish video row
 200                    NEWUNPACK(I)=15 ;set pixels to white
 300            K=K+128                 ;jump to next row
                CALL REPACK(2048,NEWUNPACK,NEWPACKED)
                M2=M1/2         ;write one-half the #of blocks read
                CALL WRBLK(2,M2,NEWPACKED,8,IER)
 400            IF(IER.NE.1)TYPE"WRBLK #",M2," error:",IER
C
C***********************************************************************
C
C       Set the rest of the picture to white (15) for the most
C       aesthetic display.  These pixels are later set to zero
C       by the program NORMALIZE.
C
        DO 500 J=1,4096
 500            OLDPACKED(J)=177777K
        DO 600 J=32,48,16
                CALL WRBLK(2,J,OLDPACKED,16,IER)
 600            IF(IER.NE.1)TYPE"WRBLK #",J," error:",IER
        CALL TIMER(1)           ;stop timer
        WRITE(10,700)OUTFILE(1)
 700    FORMAT(" The reduced picture is in the file --->  ",S13)
        CALL RESET
        STOP
        END
C
C************** Program REDUCE *******************************************
```

APPENDIX E:

CORRELATION IMPLEMENTATION

This appendix contains the following programs:

1. NORMALIZE

2. CMULTIPLY

3. IMULTIPLY

```
C*********************************************************************
C
C       Program NORMALIZE               Written by Lt. Jim Cromer
C       Fortran 5
C
C       If switch 'U' is chosen:
C       This program normalizes the upper left quadrant of an
C       input packed video file into twenty-four (30 column by
C       20 row) normalized grid blocks, with the average sum
C       of the squares of the normalized pixel values per unit
C       template window area equal to one (assumes a 23x47
C       reduced template window).
C
C       If switch 'L' is chosen:
C       The entire lower right quadrant is normalized to give
C       it an energy of unity.  Switch L is usually used to
C       normalize templates.
C
C       Otherwise:
C       The program will ask the user to input the number
C       of horizontal and vertical grid rectangles.
C       Choose from:
C       Horizontal ---> 1,2,3,4,5,6,8,10,
C                       12,15,20,24,30,40,60,120
C       Vertical   ---> 1,2,3,5,6,10,15,30
C
C       The output file is a 256x256  element
C       complex contiguous file (or random, if a contiguous file
C       cannot be created).  The program assumes the input file
C       is a reduced picture in the upper left or lower right hand
C       quadrants.  It is normally used in the sequence
C           -> REDUCE NORMALIZE DIRECT CMULTIPLY INVERSE  <-
C
C       Execution Line Format:
C               NORMALIZE[/U or L] infile outfile
C         One of the switches must be selected for use as an
C         automatic program, as in a macro file; a U indicates
C         to normalize the upper left-hand quadrant; an L
C         indicates that the lower right-hand quadrant is to
C         be normalized.  Pixel values outside of the selected
C         quadrant will be set to zero.
C
C       Load  Line Format:
C               RLDR NORMALIZE IOF TIMER UNPACK @FLIB@
C
C*********************************************************************
        REAL NORM(120,30),ENERGY(120,30)
        INTEGER ARRAY(256),INFILE(7),OUTFILE(7),ROW(32,4,2)
        INTEGER ONECOUNT,TWOCOUNT,MS(2),UNPACKED(512),MAIN(7)
        INTEGER VGRID,HGRID,GRDBLK
        INTEGER HWIDTH,VNUMOFG,HNUMOFG,BPERGRID
        COMPLEX CNORM(1024),CZERO
        LOGICAL TEST
        EQUIVALENCE (NORM,ENERGY)
```

```
      ISTART=0
      ISTOP=1
C
C
C****** I/O FILE MANAGEMENT ********************************************
C
      CALL RESET
      CALL IOF(2,MAIN,INFILE,OUTFILE,IDUM,MS,I2,I3,I4)
      CALL DFILW("TEMP",IER)
      IF(IER.NE.1.AND.IER.NE.13)TYPE"TEMP DFILW error:",IER
      CALL CFILW("TEMP",2,IER)
      IF(IER.NE.1)TYPE"TEMP CFILW error:",IER
      IER=1
      JER=1
      CALL OPEN(2,OUTFILE,2,IER)
      IF(IER.EQ.1)GO TO 55
      TYPE"Attempting to create a contiguous file."
      IF(IER.EQ.13)CALL CFILW(OUTFILE,3,1024,JER)
      IF(JER.NE.41)TYPE"Successfully created a contiguous file."
      IF(JER.EQ.41)TYPE"Must create a random file instead."
      IF(JER.EQ.41)CALL CFILW(OUTFILE,2,JER)
      IF(JER.NE.1)TYPE"OUTFILE CFILW error:",IER
      IF(IER.EQ.13)CALL OPEN(2,OUTFILE,3,IER)
      IF(IER.NE.1)TYPE"OUTFILE OPEN ERROR:",IER
 55   CALL OPEN(0,INFILE,1,IER)
      IF(IER.NE.1)TYPE"INFILE OPEN error:",IER STOP
      CALL OPEN(1,"TEMP",2,IER)
      IF(IER.NE.1)TYPE"TEMP OPEN error:",IER
C
C
C****** DETERMINE SWITCHES AND SET VARIABLES ******************
C
      IF(MS(1).EQ.16)GO TO 500          ;switch was L
      IF(MS(2).EQ.2048)GO TO 1          ;switch was U
      GO TO 556
 5555 TYPE"Input error!<7> 0< input <121","<15>"
 556  ACCEPT"Enter horizontal # of grid rectangles
     $ (1-120): ",HNUMOFG
      TEST=HNUMOFG.LT.1.OR.HNUMOFG.GT.120
      IF(TEST)GO TO 5555
      IHOLD=MOD(120,HNUMOFG)
      IF(IHOLD.NE.0)TYPE"Try again.  Input must
     $ divide evenly into 120."
      IF(IHOLD.NE.0)GO TO 556
 558  ACCEPT"Enter vertical # of grid rectangles
     $ (1-30): ",VNUMOFG
      IHOLD=MOD(30,VNUMOFG)
      TEST=VNUMOFG.LT.1.OR.VNUMOFG.GT.30
      IF(TEST)TYPE"Out of range!<7><15>"
      IF(TEST)GO TO 558
      IF(IHOLD.NE.0)TYPE"Try again.  Input must divide
     $ evenly into 30."
      IF(IHOLD.NE.0)GO TO 558
      GO TO 3
 500    TYPE"Lower right quadrant option on."
        IMIN=32
        IROW=2
        JMIN=0
        TWOCOUNT=512
        KSTART=128
        GO TO 2
 1      TYPE"Upper left quadrant option on."
        HNUMOFG=4
        VNUMOFG=6
 3      IROW=1
        IMIN=0
        JMIN=512
        TWOCOUNT=0
        KSTART=0
C
C
C****** CREATE THE NORMALIZED FILE *********************************
C
 2      IMAX=IMIN+31    ;limits on infile RDBLK (lower)
        JMAX=JMIN+511   ;limits on outfile zeroed blocks
        CALL TIMER(ISTART)      ;start timer
        TYPE"<15>","Creating the normalized file."
        ONECOUNT=0              ;workfile WRBLK counter
        SUMSQ=0.0
        CZERO=CMPLX(0.0,0.0)
        DO 10 J=1,1024
 10             CNORM(J)=CZERO
        DO 20 J=JMIN,JMAX,16    ;zero appropriate outfile rows
                CALL WRBLK(2,J,CNORM,16,IER)
 20             IF(IER.NE.1)TYPE"CNORM zero WRBLK error:",IER
        IF(IMIN.EQ.32)GO TO 200         ;switch was L
C
C       *** NORMALIZE UPPER QUADRANT ***
C
C       Create the unpacked workfile and determine the
C       energy content of it. Work on 4 picture rows per loop.
C
        HWIDTH=120/HNUMOFG      ;width of rectangle
        BPERGRID=30/VNUMOFG     ;RDBLKS per grid
        TLENGTH=23.0    ;length of reduced template
        TWIDTH=47.0     ;width  ''    ''      ''
        VLENGTH=4.0*FLOAT(BPERGRID)             ;4 rows/block
        TAREA=TLENGTH*TWIDTH    ;template area
        GAREA=FLOAT(HWIDTH)*VLENGTH     ;rectangle area
        AREAFACTOR=TAREA/GAREA
        DO 19 K=1,30    ;initialize energy terms
        DO 19 J=1,120
 19             ENERGY(J,K)=0.0
C
C       Determine energy in 30 blocks (120 rows)
C
        INBLOCK=IMIN-1  ;set RDBLK counter
        DO 40 VGRID=1,VNUMOFG           ;do 5 rows of grids
```

```
         DO 39 GRDBLK=1,BPERGRID         ;of 6 blocks each
                 INBLOCK=INBLOCK+1       ;RDBLK counter
                 CALL RDBLK(0,INBLOCK,ARRAY,1,IER)
                 KK=IMIN
                 DO 25 II=1,4    ;do 4 video rows
                         DO 21 J=1,2     ;set first 8 columns
                         KK=KK+1         ;to zero
                         ROW(J,II,IROW)=0 ;(noise terms)
  21                     CONTINUE
                         DO 23 J=3,32    ;arrange non-zero portion
                         KK=KK+1         ;of picture for processing
                         ROW(J,II,IROW)=ARRAY(KK)
  23                     CONTINUE
                         KK=KK+32
  25             CONTINUE
                 CALL UNPACK(128,ROW(1,1,IROW),UNPACKED)
 C
 C       Determine energy in 4 rows
 C
                 MINCOL=9
                 DO 30 HGRID=1,HNUMOFG               ;do 5 columns of grids,
                 MAXCOL=MINCOL+HWIDTH-1
                 DO 28 KCOL=MINCOL,MAXCOL
                    DO 26 JROW=0,3          ;do 4 rows
                    ENERGY(HGRID,VGRID)=ENERGY(HGRID,VGRID)+
     $                   (FLOAT(UNPACKED(KCOL+JROW*128)+1))**2
  26                CONTINUE
  28             CONTINUE
                 MINCOL=MAXCOL+1
  30             CONTINUE
         CALL WRBLK(1,ONECOUNT,UNPACKED,2,IER)
         IF(IER.NE.1)TYPE"1RDBLK #",ONECOUNT," error:",IER
         ONECOUNT=ONECOUNT+2     ;WRBLK counter
  39     CONTINUE        ;do next block of grids
  40     CONTINUE        ;do next grid row
 C       IF(HNUMOFG.GT.5)GO TO 666
 C               WRITE(12,3000)INFILE(1)
 C 3000          FORMAT(" ENERGIES OF ",S13,///)
 C       DO 100 VGRID=1,VNUMOFG
 C               WRITE(12,2000)VGRID,(ENERGY(HGRID,VGRID),HGRID=1,HNUMOFG
 C 2000          FORMAT(" GRID ROW",I2,5(10X,F12.2),/)
 C 100   CONTINUE
 C
 C       Determine the normalization factors
 C
  666    DO 50 VGRID=1,VNUMOFG                       ;vertical
                 DO 45 HGRID=1,HNUMOFG               ;horizontal
                 IF(ENERGY(HGRID,VGRID).LE.1.0)ENERGY(HGRID,VGRID)=1.0
                 NORM(HGRID,VGRID)=SQRT(ENERGY(HGRID,VGRID)*AREAFACTOR)
  45             CONTINUE
  50     CONTINUE
 C
 C       Normalize and create the output file
 C
```

```
      INBLOCK=-2      ;RDBLK counter
      DO 70 VGRID=1,VNUMOFG
      DO 60 GRDBLK=1,BPERGRID
        INBLOCK=INBLOCK+2
        CALL RDBLK(1,INBLOCK,UNPACKED,2,IER)
        MINCOL=9      ;starting columnn
        ICNT=0
        DO 58 HGRID=1,HNUMOFG         ;do portion of 5 grid blocks
           MAXCOL=MINCOL+HWIDTH-1  ;width=24
           DO 56 KCOL=MINCOL,MAXCOL         ;do width of 1
                                            ;grid block
           ICNT=ICNT+1
           DO 54 JROW=0,3
             OUTPUT=FLOAT(UNPACKED(ICNT+JROW*128)+1)
    $        /NORM(HGRID,VGRID)
             CNORM(KCOL+JROW*256)=CMPLX(OUTPUT,0.0)
 54        CONTINUE
 56        CONTINUE
           MINCOL=MAXCOL+1
 58     CONTINUE
        CALL WRBLK(2,TWOCOUNT,CNORM,16,IER)
        IF(IER.NE.1)TYPE"2WRBLK #",TWOCOUNT," error:",IER
        TWOCOUNT=TWOCOUNT+16
 60   CONTINUE
 70   CONTINUE
      DO 80 J=1,1024
           CNORM(J)=CZERO
 80   CONTINUE
      DO 90 J=1,2     ;zero out noise rows
           CALL WRBLK(2,TWOCOUNT,CNORM,16,IER)
           IF(IER.NE.1)TYPE"2WRBLK #",TWOCOUNT," error:",IER
           TWOCOUNT=TWOCOUNT+16
 90   CONTINUE
      GO TO 555
C
C     **** Normalize lower quadrant (template) ****
C
 200  DO 240 I=IMIN,IMAX      ;read non-zero portion of infile
           CALL RDBLK(0,I,ARRAY,1,IER)
           IF(IER.NE.1)TYPE"0RDBLK #",I," error:",IER
C
C     Set nonzero portion of ARRAY equal to ROW
C
           KK=IMIN
           DO 225 II=1,4
                DO 223 J=1,32
                KK=KK+1
 223            ROW(J,II,IROW)=ARRAY(KK)
 225       KK=KK+32
C
           CALL UNPACK(128,ROW(1,1,IROW),UNPACKED)
           DO 230 J=1,512  ;determine energy in 4 rows
 230            SUMSQ=SUMSQ+UNPACKED(J)**2
           CALL WRBLK(1,ONECOUNT,UNPACKED,2,IER)
```

```
                IF(IER.NE.1)TYPE"1WRBLK #",ONECOUNT," error:",IER
 240            ONECOUNT=ONECOUNT+2
C         *******************************************************
        TENERGY=SQRT(SUMSQ)     ;the normalizing factor
C         *******************************************************
C
C
C     * Normalize the significant pixels of INFILE ****************
C
        DO 270 I=0,62,2
        KK=KSTART               ;starting column of nonzero outfile
        ICNT=0
        CALL RDBLK(1,I,UNPACKED,2,IER)
        IF(IER.NE.1)TYPE"1RDBLK #",I," error:",IER
        DO 260 K=1,4            ;normalize infile
                DO 250 J=1,128
                KK=KK+1
                ICNT=ICNT+1
                OUTPUT=UNPACKED(ICNT)/TENERGY
 250            CNORM(KK)=CMPLX(OUTPUT,0.0)
 260    KK=KK+128
        CALL WRBLK(2,TWOCOUNT,CNORM,16,IER)
        IF(IER.NE.1)TYPE"2WRBLK #",TWOCOUNT," error:",IER
 270    TWOCOUNT=TWOCOUNT+16
C
C
C*******************************************************************
C
 555    CALL TIMER(ISTOP)       ;stop timer
        WRITE(10,1000)OUTFILE(1)
 1000   FORMAT(" The normalized file is in ---> ",S13)
        CALL RESET
        CALL DFILW("TEMP",IER)
        IF(IER.NE.1)TYPE"TEMP DFILW error:",IER
        STOP
        END
C
C************ Program NORMALIZE *************************************
```

```
C***********************************************************************
C
C       Program CMULTIPLY                Written by Lt. Jim Cromer
C       Fortran 5
C
C       This program performs a point-by-point complex multi-
C       plication between INFILE1 and CONJG(INFILE2).  The input
C       files must be 256x256 point complex arrays; the output
C       file is a 256x256 point complex array.
C
C       Execution Line Format:
C               CMULTIPLY infile1 infile2 outfile
C
C       Load Line Format:
C               RLDR CMULTIPLY IOF @FLIB@
C
C***********************************************************************
        INTEGER FILE(7),INFILE1(7),INFILE2(7),MAIN(7)
        COMPLEX MA1(2048),MA2(2048),MA3(2048)
        COMMON FILE,INFILE1,INFILE2,MAIN
C
C******** I/O FILE MANAGEMENT ******************************************
C
        CALL IOF(3,MAIN,INFILE1,INFILE2,FILE,MS,I2,I3,I4)
        WRITE(10,1999)INFILE1(1),INFILE2(1),FILE(1)
 1999   FORMAT(" IN1=",S13," IN2=",S13," OUT=",S13)
        CALL OPEN(1,FILE,2,IER)
        IF(IER.EQ.1)GO TO 55
        CALL CFILW(FILE,3,1024,IER)
        IF(IER.EQ.41)CALL CFILW(FILE,2,IER)
        IF(IER.NE.1)TYPE"OUTFILE CFILW error #",IER
        CALL OPEN(1,FILE,2,IER)
        IF(IER.NE.1)TYPE"OUTFILE OPEN error #",IER
 55     CALL OPEN(2,INFILE1,2,IER)
        IF(IER.NE.1)TYPE"INFILE1 OPEN error #",IER
        CALL OPEN(3,INFILE2,2,IER)
        IF(IER.NE.1)TYPE"INFILE2 OPEN error #",IER
C
C******** Perform point-by-point multiplication ************************
C
        DO 30 I=0,992,32                 ;process 1024 blocks
                CALL RDBLK(2,I,MA2,32,IER)       ;read 8 complex rows
                IF(IER.NE.1)TYPE"2RDBLK #",I," error:",IER
                CALL RDBLK(3,I,MA3,32,IER)
                IF(IER.NE.1)TYPE"3RDBLK #",I," error:",IER
                DO 20 K=1,2048
                        MA1(K)=MA2(K)*CONJG(MA3(K))
 20             CONTINUE
                CALL WRBLK(1,I,MA1,32,IER)
                IF(IER.NE.1)TYPE"WRBLK #",I," error:",IER
 30     CONTINUE
        WRITE(10,40)FILE(1)
 40     FORMAT(" ",S13," created by CMULTIPLY")
        END
C************ Program CMULTIPLY **************************************
```

```
C***********************************************************************
C
C       Program CTOI                          Written by Lt. Jim Cromer
C       Fortran 5
C
C       This program converts a 256x256 complex file into a
C       256x256 integer file.  The real part only of the complex
C       file is saved; the imaginary part is assumed to be zero.
C       The maximum value and its position are written to first
C       3 words of block 255.  [Values greater then 2 are set
C       to 1.  All other values are divided by two.].
C
C       Execution Line Format:
C               CTOI    complex infile     integer outfile
C
C       Load Line Format:
C               RLDR CTOI IOF TIMER @FLIB@
C
C***********************************************************************
        INTEGER MAIN(7),INFILE(7),OUTFILE(7),MS([illegible]
        INTEGER OUTINTEGER(4096),COLUMNNUMBER,ROWNUMBER
        COMPLEX INCOMPLEX(4096)
C
C******** I/O FILE MANAGEMENT ****************************************
C
        CALL IOF(2,MAIN,INFILE,OUTFILE,IDUM,MS,IS1,IS2,IS3)
        CALL TIMER(0)           ;start timer
        CALL OPEN(0,INFILE,1,IER)
        IF(IER.NE.1)TYPE"INFILE OPEN error #",IER
        CALL OPEN(1,OUTFILE,3,IER)
        IF(IER.EQ.1)GO TO 1
        CALL CFILW(OUTFILE,3,256,IER)
        IF(IER.EQ.1)TYPE"Created a contigous output file."
        IF(IER.EQ.41)CALL CFILW(OUTFILE,2,IER)
        IF(IER.NE.1)TYPE"OUTFILE CFILW error #",IER
        CALL OPEN(1,OUTFILE,3,IER)
        IF(IER.NE.1)TYPE"OUTFILE error #",IER
C
C********* CONVERT COMPLEX WORDS TO INTEGER **********************
C
 1      IMAX=0
        DO 20 J=0,960,64
                CALL RDBLK(0,J,INCOMPLEX,64,IER)
                IF(IER.NE.1)TYPE"INCOMPLEX RDBLK #",J," error:",IER
                DO 10 K=1,4096
                        AREAL=REAL(INCOMPLEX(K))/2.0
                        OUTINTEGER(K)=INT(AREAL*32767.0)
                        IF(AREAL.GE.1.0)OUTINTEGER(K)=32767
                        IF(OUTINTEGER(K).LT.IMAX)GO TO 10
                        COLUMNNUMBER=K-(INT((K-1)/256)*256)
                        ROWNUMBER=INT((K-1)/256)+1+J/4
                        IMAX=OUTINTEGER(K)
 10             CONTINUE
 17             CALL WRBLK(1,(J/4),OUTINTEGER,16,IER)
```

```
                IF(IER.NE.1)TYPE"OUTINTEGER WRBLK #",
     $(J/2)," error #",IER
 20     CONTINUE
        OUTINTEGER(1)=IMAX
        OUTINTEGER(2)=COLUMNNUMBER
        OUTINTEGER(3)=ROWNUMBER
        CALL WRBLK(1,255,OUTINTEGER,1,IER)
        IF(IER.NE.1)TYPE"IMAX WRBLK error:",IER
        CALL TIMER(1)             ;stop timer
        TYPE" The maximum integer value is:",IMAX
        TYPE"                     @ ROWNUMBER=",ROWNUMBER
        TYPE"                     @ COLUMN    =",COLUMNNUMBER
        RMAX=(FLOAT(IMAX))/32767.0
        TYPE"                    IMAX/32767  =",RMAX
        WRITE(10,1000)INFILE(1),OUTFILE(1)
 1000   FORMAT(" The complex file ",S8," has been
     $converted to the integer file ",S8)
        CALL RESET
        STOP
        END
C
C********* Program CTOI ***********************************************
```

```
C*********************************************************************
C
C       Program IMULTIPLY       Written by Lt. Jim Cromer
C
C       This program calculates the point-by-point geometric
C       mean between INFILE1 and INFILE2; the product is output
C       to OUTFILE.  All files must be 256x256 integer files.
C
C       Execution Line Format:
C               IMULTIPLY infile1 infile2 outfile
C
C       Load Line Format:
C               RLDR IMULTIPLY IOF @FLIB@
C
C*******************************************************************
        INTEGER MAIN(7),INFILE1(7),INFILE2(7),OUTFILE(7)
        INTEGER GS(2),LS1(2),LS2(2),LS3(2)
        INTEGER FACTOR1(8192),FACTOR2(8192),PRODUCT(8192)
C
C******** I/O FILE MANAGEMENT ************************
C
        CALL IOF(3,MAIN,INFILE1,INFILE2,OUTFILE,GS,LS1,LS2,LS3)
        JER=1
        KER=1
        CALL OPEN(1,INFILE1,2,IER)
        IF(IER.NE.1)STOP" INFILE1 OPEN ERROR"
        CALL OPEN(2,INFILE2,2,IER)
        IF(IER.NE.1)STOP"INFILE2 OPEN ERROR"
C
C       First check to see if the file exists; if it
C       doesn't, try to create a contiguous file.
C
        CALL OPEN(3,OUTFILE,2,IER)
        LER=IER
        IF(IER.EQ.13)CALL CFILW(OUTFILE,3,256,JER)
        IF(JER.NE.1)CALL CFILW(OUTFILE,2,KER)
        IF(KER.NE.1)STOP"OUTFILE CFILW ERROR"
        IF(KER.EQ.1.OR.JER.EQ.1)IER=1
        IF(IER.NE.1)STOP"OUTFILE OPEN ERROR"
        IF(LER.NE.1)CALL OPEN(3,OUTFILE,2,IER)
        IF(IER.NE.1)STOP"OUTFILE OPEN ERROR"
C******* Perform point-by-point multiplication
C
C
        DO 20 I=0,224,32
                CALL RDBLK(1,I,FACTOR1,32.IER)
                IF(IER.NE.1)TYPE"1RDBLK #",I," error:",IER
                CALL RDBLK(2,I,FACTOR2,32,IER)
                IF(IER.NE.1)TYPE"2RDBLK #",I," error:",IER
                DO 10 J=1,8192
                  AHOLD=SQRT((FLOAT(FACTOR1(J))*FLOAT(FACTOR2(J))))
                  PRODUCT(J)=INT(AHOLD+0.5)
 10             CONTINUE
                CALL WRBLK(3,I,PRODUCT,32,IER)
                IF(IER.NE.1)TYPE"WRBLK  #",I," error:",IER
        TYPE"   *** DATA CRUNCH! ***<11><11>*** DATA CRUNCH! ***"
 20     CONTINUE
C
C****** Write message to the CRT terminal *********
C
        WRITE(10,1000)OUTFILE(1)
 1000   FORMAT(///,"The integer product file is named ---> ",S13,/)
        CALL RESET
        STOP
        END
C
C******** Program IMULTIPLY ***********************************************
```

APPENDIX F:

PROCESS EVALUATION

This appendix contains the following programs:

1. ITOC
2. PEAK (F1)
3. CTOV
4. DISTANCE (EUCLID)

```
C**********************************************************************
C
C       Program ITOC                        Written by Lt. Jim Cromer
C       Fortran 5
C
C       This program converts a 256x256 integer file into
C       a 256x256 complex file.
C
C       Execution Line Format:
C               ITOC/[A,E,H,N, or O] integerfile[/C and or T] complex[/M]
C        Global switches ---->  A: convert all values >0
C                               E:  convert values >80% of the maximum
C       ;values not con-        H:  convert values >1/2 maximum
C       ;verted will be         N:  convert values >90% maximum
C       ;set to zero            O:  accept other conversion value
C
C       Inputfile switch ---->  C:  "crunch" 256x256 data into 256x128
C                                   array for use with PLTTRNS contour
C                               T:  set output values equal to input
C                                   value minus the threshold
C
C       Outputfile switch ---> M:  insert maximum value into 3rd
C                                   column of every 4th row
C
C       Load Line Format:
C               RLDR ITOC IOF TIMER @FLIB@
C
C**********************************************************************
        INTEGER MAIN(7),INFILE(7),OUTFILE(7),MS(2),IS2(2),IS1(2)
        INTEGER ININTEGER(4096),COLUMN,ROW
        REAL INREAL
        COMPLEX OUTCOMPLEX(4096)
C
C******** I/O FILE MANAGEMENT ****************************************
C
        CALL IOF(2,MAIN,INFILE,OUTFILE,IDUM,MS,IS1,IS2,IS3)
        PERCENT=-9999.0
        IF(ITEST(MS(1),15).EQ.1)PERCENT=0.0      ;switch was A
        IF(ITEST(MS(1),11).EQ.1)PERCENT=0.80     ;switch was E
        IF(ITEST(MS(1),8).EQ.1)PERCENT=0.50      ;switch was H
        IF(ITEST(MS(1),2).EQ.1)PERCENT=0.90      ;switch was N
 2      IF(ITEST(MS(1),1).EQ.1)ACCEPT" Enter the
     $ threshold percent (0.0 - .99): ",PERCENT
        IF(PERCENT.LT.0.0.OR.PERCENT.GT.1.0)GO TO 2
        IF(PERCENT.EQ.-9999.0)STOP"BAD GLOBAL SWITCH"
        CALL TIMER(0)           ;start timer
        CALL OPEN(0,INFILE,1,IER)
        IF(IER.NE.1)TYPE"INFILE OPEN error #",IER
        CALL OPEN(1,OUTFILE,3,IER)
        IF(IER.EQ.1)GO TO 1
        CALL CFILW(OUTFILE,3,1024,IER)
        IF(IER.EQ.1)TYPE"Created a contiguous output file."
        IF(IER.EQ.41)CALL CFILW(OUTFILE,2,IER)
        IF(IER.NE.1)TYPE"OUTFILE CFILW error #",IER
```

```
      CALL OPEN(1,OUTFILE,3,IER)
      IF(IER.NE.1)TYPE"OUTFILE error #",IER
C
C********* CONVERT INTEGER WORDS TO COMPLEX ***************************
C
 1    CALL RDBLK(0,255,ININTEGER,1,IER)
      IF(IER.NE.1)TYPE"IMAX RDBLK ERROR: ",IER
      IMAX=ININTEGER(1)
      COLUMN=ININTEGER(2)
      ROW=ININTEGER(3)
      ITHRESH=INT(PERCENT*FLOAT(IMAX))
      JFACTOR=2
      JBLOCK=64
      INCREMENT=0
      IF(IS1(1).EQ.8192)JFACTOR=1
      IF(IS1(1).EQ.8192)JBLOCK=32
      IF(IS1(1).EQ.8192)INCREMENT=256
C
C     Create the output file
C
      DO 20 J=0,480,32
            CALL RDBLK(0,(J/2),ININTEGER,16,IER)
            IF(IER.NE.1)TYPE"ININTEGER RDBLK #",(J/2)," error:",IER
            LL=0
            KK=0
            DO 10 I=1,(8*JFACTOR)
                  DO 5 K=1,256
                  KK=KK+1
                  LL=LL+1
                  IF(IS1(2).NE.4096)GO TO 3
                  INREAL=FLOAT(ININTEGER(KK)-ITHRESH)/32767.0
                  GO TO 4
 3                INREAL=FLOAT(ININTEGER(KK))/32767.0
 4                IF(ININTEGER(KK).LT.ITHRESH)INREAL=0.0
                  OUTCOMPLEX(LL)=CMPLX(INREAL,0.0)
  5         CONTINUE
            KK=KK+INCREMENT
 10         CONTINUE
C
C     Prepare output for PLTTRNS row plot
C
            IF(IS2(1).EQ.8.AND.IS1(2).EQ.4096)OUTCOMPLEX(3)
    $=CMPLX(FLOAT(IMAX-ITHRESH),0.0)
            IF(IS2(1).EQ.8.AND.IS1(2).NE.4096)
    $OUTCOMPLEX(3)=CMPLX(FLOAT(IMAX),0.0)
            CALL WRBLK(1,(J*JFACTOR),OUTCOMPLEX,JBLOCK,IER)
            IF(IER.NE.1)TYPE"OUTCOMPLEX WRBLK #",(J*JFACTOR),
    $" error :",IER
 20   CONTINUE
C
C     The data will be compressed to the front half of the
C     output plane.
C
      IF(IS1(1).NE.8192)GO TO 35
```

```
        DO 23 I=1,3
                OUTCOMPLEX(I+1792)=CMPLX(0.0,0.0)
 23     CONTINUE
        CALL WRBLK(1,480,OUTCOMPLEX,32,IER)
        IF(IER.NE.1)TYPE"WRBLK (DATA) #480 error:",IER
        DO 25 I=1,4096
                OUTCOMPLEX(I)=CMPLX(0.0,0.0)
 25     CONTINUE
        DO 30 J=512,960,64
                CALL WRBLK(1,J,OUTCOMPLEX,64,IER)
                IF(IER.NE.1)TYPE"WRBLK #",J," error:",IER
 30     CONTINUE
        GO TO 40
C
C       Expanded output only
C
 35     DO 38 I=1,3
                OUTCOMPLEX(I+3840)=CMPLX(0.0,0.0)
 38     CONTINUE
        CALL WRBLK(1,960,OUTCOMPLEX,64,IER)
        IF(IER.NE.1)TYPE"WRBLK (DATA) #960 error:",IER
C
C       Send completion message to CRT
C
 40     CALL TIMER(1)           ;stop timer
        TYPE"IMAX=",IMAX
        TYPE"ITHRESH=",ITHRESH
        TYPE"PERCENT=",PERCENT
        TEMP=FLOAT(ITHRESH)/32767.0
        TYPE" Normalized threshold=",TEMP
        WRITE(10,1000)INFILE(1),OUTFILE(1)
 1000   FORMAT(" The integer file ",S8," has been
     $converted to the complex file ",S8)
        CALL RESET
        STOP
        END
C
C********* Program ITOC **************************************************
```

```
C**********************************************************************
C
C       Program PEAK                        Written by Lt. Jim Cramer
C       Fortran 5
C
C       This program searches a 256x256 integer file for isolated
C       regions of which all values are above a given threshold.
C       If the input array is thought of as a 3-dimensional sur-
C       face, then these regions will be the "peaks" of the surface.
C       The position and value of both local and global peaks is
C       determined.
C
C       Execution Line Format:
C               PEAK
C
C       Load Line Format:
C               RLDR PEAK F1 @FLIB@
C
C       Flag Values:
C               PEAK            --->  -1 - peak closed
C               ;holds condition       0 - peak unused
C               ;of global peaks       1 - peak open
C
C               STATUS          --->  -1 - row peak matched
C               ;holds condition       0 - unused in current row
C               ;of row peaks          1 - row peak unmatched
C
C               INSIDE          --->  .TRUE. - previous value
C               ;determines if                 tested > threshold
C               ;pointer in           .FALSE. - else
C               ;interior or exterior
C               ;of a row peak
C
C**********************************************************************
        REAL NORMALIZE(10)
        INTEGER INFILE(7),WIDTH(10),PCENTMAX(10),LENGTH(10)
        INTEGER VALUE(256),F1,THRESH,ISTART(10),ISTOP(10)
        INTEGER ROW,COLUMN,FVALUE,TEMPPEAK,PROW(10),IMAX(10)
        INTEGER MAXCOLUMN(10),PSTOP(10,256),PSTART(10,256)
        INTEGER PVALUE(10),PCOLUMN(10),STATUS(10),PEAK(10)
        INTEGER JMAX(10),JMIN(10),RANK(10),IRANK(0:10)
        LOGICAL ATEST,INSIDE
C
C******** INITIALIZE VARIABLES ***********************************
C
        IOPT=0
  100   DO 200 I=1,10
                IRANK(I)=0
                RANK(I)=0
                PEAK(I)=0
                PVALUE(I)=0
                PCOLUMN(I)=0
                PROW(I)=0
                DO 150 J=1,256
```

```
                        PSTART(I,J)=257
                        PSTOP(I,J)=0
  150           CONTINUE
  200   CONTINUE
C
C******** SET I/O PARAMETERS *****************************************
C
        IF(IOPT.EQ.2)GO TO 250
        ACCEPT"What is the name of the input integer file? "
        READ(11,1000)INFILE(1)
 1000   FORMAT(S13)
        CALL OPEN(0,INFILE,2,IER)
        IF(IER.NE.1)STOP"INFILE OPEN ERROR"
        CALL RDBLK(0,255,VALUE,1,IER)
        MAXIMUM=VALUE(1)
        TYPE"Absolute max=",MAXIMUM
        GO TO 250
  210   TYPE"Input error.  Percentage must be between 1-100.<7><15>"
  250   ACCEPT"Enter integer percentage of absolute maximum
     $ to be included: ",IPERCENT
        IF(IPERCENT.LT.1.OR.IPERCENT.GT.100)GO TO 210
        THRESH=INT(FLOAT(IPERCENT)*FLOAT(MAXIMUM)/100.0)
        TYPE"INTEGER THRESHOLD=",THRESH
C
C*******************************************************************
C
C       Loop through the scan and matching modules 254 times
C       (test all but first and last rows)
C
        DO 600 ROW=2,255        ;test rows 2-255
                CALL RDBLK(0,(ROW-1),VALUE,1,IER)
                IF(IER.NE.1)TYPE"RDBLK #",(ROW-1)," error:",IER
                INSIDE=.FALSE.
                NUMPEAKS=0
                DO 300 I=1,10
                        STATUS(I)=0
  300           CONTINUE
C
C******** SCAN ROW ***************************************************
C
C
C       This module determines if the scanning pointer is in
C       the interior or the exterior of a row peak.  When a
C       row peak is encountered, the peak counter NUMPEAKS is
C       increased, and the flag STATUS is set to the unmatched
C       condition.  The maximum value and corresponding column
C       number for each row peak is stored.
C
        DO 350 COLUMN=2,255
                FVALUE=F1(VALUE(COLUMN-1),VALUE(COLUMN),VALUE(COLUMN+1))
                IF(FVALUE.LT.THRESH.AND..NOT.INSIDE)GO TO 350
                IF(FVALUE.GE.THRESH.AND.INSIDE)GO TO 320
                IF(FVALUE.GE.THRESH.AND..NOT.INSIDE)GO TO 310
                IF(FVALUE.LT.THRESH.AND.INSIDE)GO TO 330
  310           NUMPEAKS=NUMPEAKS+1     ;row-peak counter
                STATUS(NUMPEAKS)=1      ;row peak opened,unmatched
                IMAX(NUMPEAKS)=0
                ISTOP(NUMPEAKS)=256
                ISTART(NUMPEAKS)=COLUMN
                INSIDE=.TRUE.
  320           IF(IMAX(NUMPEAKS).GT.FVALUE)GO TO 350
                        IMAX(NUMPEAKS)=FVALUE
                        MAXCOLUMN(NUMPEAKS)=COLUMN
                GO TO 350
  330           INSIDE=.FALSE.
                ISTOP(NUMPEAKS)=COLUMN-1
 350    CONTINUE
C
C
C
C       If no values above threshold were found in the
C       last row tested, then close all open peaks.
C
                IF(NUMPEAKS.NE.0)GO TO 400
                DO 370 I=1,10
                        IF(PEAK(I).NE.1)GO TO 370
                        PEAK(I)=-1
                        JMAX(I)=ROW-1
 370            CONTINUE
                GO TO 600
C
C
 400            CONTINUE
C
C******** ATTEMPT TO MATCH *******************************************
C
C       Attempt to match row peaks to open global peaks.
C
        DO 500 I=1,10
                IF(PEAK(I).NE.1)GO TO 500  ;check all open peaks
                PEAK(I)=-1      ;will be closed unless matched
                JMAX(I)=ROW-1
                DO 450 TEMPPEAK=1,NUMPEAKS
                        IF(STATUS(TEMPPEAK).EQ.0)GO TO 450
                        ATEST=PSTOP(I,(ROW-1)).LT.ISTART(TEMPPEAK)
     $                  .OR.PSTART(I,(ROW-1)).GT.ISTOP(TEMPPEAK)
                        IF(ATEST)GO TO 450      ;did not match
                        PEAK(I)=1
                        STATUS(TEMPPEAK)=-1
                        PSTART(I,ROW)=MIN(PSTART(I,ROW),ISTART(TEMPPEAK)
                        PSTOP(I,ROW)=MAX(PSTOP(I,ROW),ISTOP(TEMPPEAK))
                        IF(PVALUE(I).GE.IMAX(TEMPPEAK))GO TO 450
                        PVALUE(I)=IMAX(TEMPPEAK)
                        PROW(I)=ROW
                        PCOLUMN(I)=MAXCOLUMN(TEMPPEAK)
 450            CONTINUE
 500    CONTINUE
C
```

```
C******** MUST OPEN A NEW GLOBAL PEAK ************************************
C
C       Match unmatched row peaks to unused global peaks.
C
        DO 550 TEMPPEAK=1,NUMPEAKS
                IF(STATUS(TEMPPEAK).NE.1)GO TO 550
                DO 510 I=1,10
                IF(STATUS(TEMPPEAK).NE.1)GO TO 510
                IF(PEAK(I).NE.0)GO TO 510
                        STATUS(TEMPPEAK)=-1
                        PEAK(I)=1
                        JMIN(I)=ROW
                        JMAX(I)=ROW
                        PSTOP(I,ROW)=ISTOP(TEMPPEAK)
                        PSTART(I,ROW)=ISTART(TEMPPEAK)
                        PVALUE(I)=IMAX(TEMPPEAK)
                        PROW(I)=ROW
                        PCOLUMN(I)=MAXCOLUMN(TEMPPEAK)
                        TYPE"PEAK #",I," START ROW:",ROW
 510            CONTINUE
 550    CONTINUE
 600    CONTINUE
C
C******** ELIMINATE MULTIPLY DEFINED PEAKS *******************
C
        DO 650 I=1,9
                IF(PEAK(I).EQ.0)GO TO 650
                DO 620 J=I,10
                        IF(PEAK(J).EQ.0)GO TO 620
                        IF(I.EQ.J)GO TO 620
                        IF(PVALUE(J).NE.PVALUE(I))GO TO 620
                        IF(PROW(J).NE.PROW(I))GO TO 620
                        IF(PCOLUMN(J).NE.PCOLUMN(I))GO TO 620
                        TYPE"*** MULTIPLY DEFINED PEAK FOUND ***"
                        PEAK(J)=999
                        JMIN(I)=MIN(JMIN(I),JMIN(J))
                        JMAX(I)=MAX(JMAX(I),JMAX(J))
                        JSTART=JMIN(I)
                        JSTOP=JMAX(I)
                        DO 610 ROW=JSTART,JSTOP
                        PSTOP(I,ROW)=MAX(PSTOP(I,ROW),PSTOP(J,ROW))
                        PSTART(I,ROW)=MIN(PSTART(I,ROW),PSTART(J,ROW))
 610                    CONTINUE
 620            CONTINUE
 650    CONTINUE
        DO 700 I=1,10
                IF(PEAK(I).EQ.999)PVALUE(I)=0
 700    CONTINUE
C
C******** SORT PEAKS ACCORDING TO THEIR MAXIMUM VALUES *********
C
        DO 706 K=1,10
                LCOUNT=0
                DO 704 I=1,10
                IF(PEAK(I).EQ.0)GO TO 704
                IF(PVALUE(I).EQ.32767)PVALUE(I)=32766
                ICOUNT=0
                DO 702 J=1,10
                        IF(PEAK(J).EQ.0)GO TO 702
                        IF(I.EQ.J)GO TO 702
                        ATEST=PVALUE(I).EQ.PVALUE(J)
                        IF(.NOT.ATEST)GO TO 702
                        LCOUNT=LCOUNT+1
                        ICOUNT=ICOUNT+1
                        PVALUE(J)=PVALUE(J)-ICOUNT
702             CONTINUE
704             CONTINUE
                IF(LCOUNT.EQ.0)GO TO 708
706     CONTINUE
708     IRANK(0)=32767
        DO 720 I=1,10
                IF(PEAK(I).EQ.0)GO TO 720
                DO 710 J=1,10
                        IF(PEAK(J).EQ.0)GO TO 710
                        IF(PVALUE(J).GE.IRANK(I-1))GO TO 710
                        IRANK(I)=MAX(IRANK(I),PVALUE(J))
710             CONTINUE
720     CONTINUE
        DO 800 I=1,10
                IF(PEAK(I).EQ.0)GO TO 800
                DO 750 J=1,10
                        IF(IRANK(J).EQ.0)GO TO 750
                        IF(PVALUE(I).EQ.IRANK(J))RANK(I)=J
750             CONTINUE
800     CONTINUE
C
C******** COMPUTE OUTPUT VALUES *********************************
C
810     DO 830 I=1,10
                IF(PEAK(I).EQ.999)PEAK(I)=0
                IF(PEAK(I).EQ.0)GO TO 830
                PCENTMAX(I)=INT(PVALUE(I)/(0.01*MAXIMUM)+0.5)
                NORMALIZE(I)=FLOAT(PVALUE(I))/32767.0
                LENGTH(I)=JMAX(I)-JMIN(I)+1
                WIDTH(I)=0
                JSTART=JMIN(I)
                JSTOP=JMAX(I)
                DO 820 J=JSTART,JSTOP
                        IHOLD=PSTOP(I,J)-PSTART(I,J)+1
                        WIDTH(I)=MAX(IHOLD,WIDTH(I))
820             CONTINUE
830     CONTINUE
850     ICH=10
870     IF(ICH.EQ.12)WRITE(12,1500)
1500    FORMAT(////)
        WRITE(ICH,2000)
        IF(ICH.EQ.10)WRITE(10,3000)INFILE(1)
        IF(ICH.EQ.12)WRITE(12,4000)INFILE(1)
        RTHRESH=FLOAT(THRESH)/32767.0
        WRITE(ICH,5000)RTHRESH,IPERCENT
        WRITE(ICH,6000)
        WRITE(ICH,7000)
        WRITE(ICH,8000)
 2000   FORMAT(//,15X,"  ************************************************
    $*****************")
 3000   FORMAT(//,30X,"INTEGER FILE EVALUATED  ---> ",S13,//)
 4000   FORMAT(//,30X,"INTEGER FILE EVALUATED  ---> <10>",S13,//)
 5000   FORMAT(40X,"THRESHOLD=",F5.3,/36X,"% OF MAX PEAK:",I5,//)
 6000   FORMAT(21X,"PEAK   %MAX",33X,"NORMALIZED")
 7000   FORMAT(21X,"  #    PEAK   ROW   COLUMN  WIDTH  LENGTH
    $PVALUE")
 8000   FORMAT(21X,"----   ----   ---   ------  -----  ------
    $--------")
        DO 910 I=1,10
                DO 900 J=1,10
                IF(RANK(J).NE.I)GO TO 900
                WRITE(ICH,9000)I,PCENTMAX(J),PROW(J),PCOLUMN(J),WIDTH(J)
    $LENGTH(J),NORMALIZE(J)
 9000           FORMAT(15X,I9,I7,I7,I8,I7,I7,F12.3)
C 9000           FORMAT(19X,I5,2X,I5,2X,I5,2X,I6,1X,I6,2X,I5,1X,F11.3)
 900            CONTINUE
 910    CONTINUE
        WRITE(ICH,2000)
        IF(ICH.EQ.12)GO TO 950
        ACCEPT"Enter a 1 to send results to the lineprinter: ",I
        IF(I.NE.1)GO TO 950
        ICH=12
        GO TO 870
 950    TYPE"<15>","What next?"
        TYPE"<15>"," Here are the options:<15>"
        TYPE"<15>","<11>1 — Try a new input file"
        TYPE"<15>","<11>2 — Try another threshold value"
        TYPE"<15>","<11>3 — STOP<15>"
 970    ACCEPT"Enter option ---> ",IOPT
        I=IOPT
        IF(I.LT.1.OR.I.GT.3)TYPE"<15>","Input error<7><7>!<15>"
        IF(I.LT.1.OR.I.GT.3)GO TO 970
        IF(I.EQ.2)GO TO 100
        CALL RESET
        IF(I.EQ.1)GO TO 100
        TYPE"<15>","*** EXITING PROGRAM PEAK ***<15>"
        STOP
        END
C
C********** Program PEAK ***********************************************
```

```
      INTEGER FUNCTION F1(NBEFORE,N,NAFTER)
C*************************************************************
C
C       Function F1
C
C       This function is part of the program PEAK.  In
C       PEAK, a function of N F1(N) is compared to
C       a threshold of some % of the maximum value to
C       determine if a local peak was found.
C
C*************************************************************
        F1=N
C       Other possible functions:
C
C       F1=INT(FLOAT(NBEFORE+N+NAFTER)/3.0+0.5)
C       F1=INT(FLOAT(NBEFORE+2*N+NAFTER)/4.0+0.5)
C       F1=INT(FLOAT(NBEFORE+3*N+NAFTER)/5.0+0.5)
        RETURN
        END
C
C********* Function F1 ****************************************
```

```
C***********************************************************************
C
C       Program CTOV            by Lt Jim Cromer
C       Fortran 5
C
C       This program converts a complex input file (imaginary part
C       assumed zero) into a video output file.  The input file is
C       linearly scaled to a 0-15 output range.  Minimum and
C       maximum values to be included are input by the user.
C
C       Execution Line Format:
C               CTOV
C
C       Load Line Format:
C               RLDR CTOV XWRBLK TIMER @FLIB@
C
C*************************************************************************
        REAL LOWER
        INTEGER IARRAY(1024),CINFILE(7),VOUTFILE(7)
        COMPLEX CARRAY(1024)
C
C******** I/O FILE MANAGEMENT *********************
C
        ACCEPT"Enter name of complex input file: "
        READ(11,2000)CINFILE(1)
 2000   FORMAT(S13)
        ACCEPT"Enter name of video output file: "
        READ(11,2000)VOUTFILE(1)
        CALL TIMER(0)
        CALL DFILW(VOUTFILE,IER)
        IF(IER.NE.1.AND.IER.NE.13)STOP"DFILW ERROR"
        CALL CFILW(VOUTFILE,2,IER)
        IF(IER.NE.1)STOP"CFILW ERROR"
        CALL OPEN(1,CINFILE,2,IER)
        IF(IER.NE.1)STOP"1 OPEN ERROR"
        CALL OPEN(2,VOUTFILE,2,IER)
        IF(IER.NE.1)STOP"2 OPEN ERROR"
C
C******* Determine maximum and minimum values *********
C
        RMAX=0.0
        RMIN=99999.99
        DO 2 K=0,63
                CALL RDBLK(1,(K*16),CARRAY,16,IER)
                IF(IER.NE.1)TYPE"1 RDBLK #",(K*16)," error:",IER
                DO 1 J=1,1024
                        A=REAL(CARRAY(J))
                        RMAX=AMAX1(RMAX,A)
                        RMIN=AMIN1(RMIN,A)
 1              CONTINUE
                IF(MOD((K+1),4).EQ.0)TYPE"BLOCK",(K*16)," searched."
 2      CONTINUE
C
C******** Determine the linear scale to be used ***********
```

```
C
      TYPE"Maximum=",RMAX,"     Minimum=",RMIN
      ACCEPT"Enter maximum to be included:",UPPER
      ACCEPT"Enter minimum to be included:",LOWER
      SCALE=15.999/(UPPER-LOWER)
C
C********* Create the output file ****************
C
      DO 20 K=0,63
              CALL RDBLK(1,(K*16),CARRAY,16,IER)
              IF(IER.NE.1)TYPE"1 RDBLK #",(K*16)," error:",IER
              DO 10 J=1,1024
                      IARRAY(J)=15
                      A=REAL(CARRAY(J))
                      IF(A.LT.LOWER)IARRAY(J)=0
                      IF(A.GE.LOWER.AND.A.LE.UPPER)
     $        IARRAY(J)=INT((A-LOWER)*SCALE)
 10           CONTINUE
              CALL XWRBLK(2,K,IARRAY,1,IER)
              IF(IER.NE.1)TYPE"2 WRBLK #",K," error:",IER
 20   CONTINUE
C
C*********** Send completion message to CRT *********
C
      CALL TIMER(1)
      WRITE(10,1000)VOUTFILE(1)
 1000 FORMAT(" The video file created is called ",S13)
      CALL RESET
      STOP
      END
C
C********** Program CTOV *************************************************
```

```
C**********************************************************************
C
C       Program DISTANCE        Written by Lt. Jim Cromer
C       Fortran 5               16 Oct 1982
C
C`      This program accepts as input a template file window and
C       up to 10 local correlation peak positions found by the
C       program PEAK.  Three distance factors will be calculated
C       between the template window and 9 scene windows (the
C       center window corresponds to the input correlation peak,
C       the other 8 are its nearest neighbors).  The score is
C       computed as the cube root of the product of the factors.
C       If any factor is less than zero (corresponding to the
C       measure for a constant gray level input scene) it is
C       set to zero.  Results are output to the lineprinter.
C
C       FACTORS USED:
C               L1FACT=100(1-NL1/NL1MAX)
C               L2FACT=100(1-NL2/NL2MAX)
C                CFACT=100(NXY-NXYMIN)/(1-NXYMIN)
C          where       NL1 is the normalized L1 distance
C                      NL2 "  "       "       L2     "
C                      NXY "  "       "       Cross-correlation value
C                   NL1MAX is the NL1 for a constant gray level input
C                   NL2MAX "  "  NL2 "  "    "       "    "       "
C                   NXYMIN "  "  NXY "  "    "       "    "       "
C
C       Execution Line Format:
C               DISTANCE
C
C       Load Line Format:
C         RLDR DISTANCE TEST XRDBLK EUCLID @FLIB@
C
C**********************************************************************
C
        REAL NL1MAX
        REAL NL1DIST,L1DIST,NSUMSQ,NDIST(10),NDISTL1(100)
        REAL CORPEAK(100),DISTL1(10),NXYMIN,NL2MAX,NL2
        INTEGER INFILE1(7),INFILE2(7),WIDTH,TB,COMMENT(200),
     $SB1,SLS,TTOP,TB1,SB,CFACTOR(100),L2FACTOR(100),L1FACTOR(100),
     $TLS,TLEFT,SCENE(1024),TEMP(1024),SLEFT(100),PCOLUMN(10),
     $COLCENT(100),ROWCENT(100),DIST(10),CTOP(100),PROW(10),SCORE
        LOGICAL REDUCED,LTEST,SUPPRESS
        COMMON /LIST1/ SCENE,TEMP,SLS,TLS,L1DIST,NL1DIST
        COMMON /LIST2/ LENGTH,WIDTH,IROWCOUNT,SUMSQ,NSUMSQ
        COMMON /LIST3/ SL1ENERGY,TL1ENERGY,SENERGY,TENERGY
        COMMON /LIST4/ S1NORM,S2NORM,T1NORM,T2NORM,CORREL
        LTEST=.TRUE.
        NL1MAX=0.46     ;normalized L1 distance between the
                        ;template and a constant gray level

C
C
C******* I/O FILE MANAGEMENT ***********************************
```

```
C
C
  99    ACCEPT"Enter template file name ---> "
        READ(11,1000)INFILE1(1)
 1000   FORMAT(S13)
        ACCEPT"Enter scene file name ---> "
        READ(11,1000)INFILE2(1)
        CALL OPEN(1,INFILE1,1,IER)
        IF(IER.NE.1)TYPE"INFILE1 OPEN ERROR #",IER
        CALL OPEN(2,INFILE2,1,IER)
        IF(IER.NE.1)TYPE"INFILE2 OPEN ERROR #",IER
C
C
C******* ENTER WINDOW PARAMETERS *******************************
C
C       The choice is given to compare the original 256x256
C       pictures, or the reduced 128x128 versions.  Reduced scene
C       files are assumed to occupy the upper left quadrant, temp-
C       lates are assumed to occupy the lower right quadrant.
C
        REDUCED=.FALSE.
 1      ACCEPT"<15>Enter a 1 to compare original video,
     $<15><11> or a 2 to compare reduced video: ",I
        IF(I.LT.1.OR.I.GT.2)TYPE"<7>INPUT ERROR!<15>"
        IF(I.LT.1.OR.I.GT.2)GO TO 1
        IF(I.EQ.2)REDUCED=.TRUE.
        IF(LTEST)GO TO 5
 3      ACCEPT"Enter a 1 to change template window parameters: ",IOPT
        IF(IOPT.NE.1)GO TO 10
 5      ACCEPT"<15>"," Enter top row of original template
     $ window (1-256):",TTOP
        ACCEPT" Enter left column of original template
     $ window (1-256):",TLEFT
        ACCEPT" Enter width of window (1-256):",WIDTH
        ACCEPT" Enter length of window (1-256):",LENGTH
        LTEST=.FALSE.
        MTOP=TTOP
        MLEFT=TLEFT
        MWIDTH=WIDTH
        MLENGTH=LENGTH
        GO TO 10
  9     TYPE"SORRY<7>.  Number of peaks can be 1-10 only."
 10     ACCEPT"<15>","Enter # of candidate peaks: ",NUMPEAKS
        IF(NUMPEAKS.GT.10.OR.NUMPEAKS.LT.1)GO TO 9
        TTOP=MTOP
        TLEFT=MLEFT
        WIDTH=MWIDTH
        LENGTH=MLENGTH
        DO 20 II=1,NUMPEAKS
          TYPE"<15>","<15>","********* PEAK",II," ***********"
          GO TO 15
 13       TYPE"<15>","Sorry.<7> Peak row must be 1-256."
 15       ACCEPT"<15>","Enter peak row number: ",PROW(II)
          IF(PROW(II).LT.1.OR.PROW(II).GT.256)GO TO 13
          CTOP(II)=256+TTOP-2*PROW(II)
          IF(REDUCED)CTOP(II)=128+INT((TTOP+1)/2)-PROW(II)
          GO TO 19
 17       TYPE"<7>Peak column must be 1-256.  Try again."
 19       ACCEPT"<15>","Enter peak column number: ",PCOLUMN(II)
          IF(PCOLUMN(II).LT.1.OR.PCOLUMN(II).GT.256)GO TO 17
          SLEFT(II)=256+TLEFT-2*PCOLUMN(II)
          IF(REDUCED)SLEFT(II)=128+INT((TLEFT+1)/2)-PCOLUMN(II)
          ROWCENT(II)=CTOP(II)+INT(LENGTH/2)
          COLCENT(II)=SLEFT(II)+INT(WIDTH/2)
          IF(REDUCED)ROWCENT(II)=CTOP(II)+INT(LENGTH/4)
          IF(REDUCED)COLCENT(II)=SLEFT(II)+INT(WIDTH/4)
 20     CONTINUE
        N=-1
          IF(REDUCED)LENGTH=INT((LENGTH+1)/2)
          IF(REDUCED)WIDTH=INT((WIDTH+1)/2)
          IF(REDUCED)TTOP=INT((TTOP+1)/2)+128
          IF(REDUCED)TLEFT=INT((TLEFT+1)/2)+128
          ILEFT=TLEFT
          ITOP=TTOP
          IWIDTH=WIDTH
          ILENGTH=LENGTH
        SUPPRESS=.FALSE.
        ACCEPT"Enter a 1 to suppress window messages: ",I2
        IF(I2.EQ.1)SUPPRESS=.TRUE.
        TYPE"Executing  .  .  .  ."
C
C**********************************************************************
C
C       This section computes the RDBLK and EUCLID search
C       window parameters, then computes the distance measures
C       for each of the windows entered.
C
        II=10
        DO 600 JJ=1,NUMPEAKS
        DO 600 J=1,3                    ;DO 9 windows
        DO 600 K=1,3                    ;compute the window shift
                II=II+1
                COLCENT(II)=COLCENT(JJ)-2+K
                CTOP(II)=CTOP(JJ)-2+J
                ROWCENT(II)=ROWCENT(JJ)-2+J
                SLEFT(II)=SLEFT(JJ)-2+K
                TL1ENERGY=0.0   ;initialize energies
                SL1ENERGY=0.0
                SENERGY=0.0
                TENERGY=0.0
C
C       The calls to TEST check to see if the input parameters are
C       legal, and modifies them if necessary:
C               0 < TOP < 257,    (TOP + LENGTH) < 258
C               0 < LEFT < 257,   (LEFT + WIDTH) < 258
C
        CALL TEST(TTOP,TLEFT)
        CALL TEST(CTOP(II),SLEFT(II))
```

```
C
C       Set RDBLK and EUCLID parameters
C
 40     N=N*-1              ;if  N=1, compute the energies
                            ;if N=-1, compute the distances
        IROWCOUNT=0
        TB=INT(FLOAT(TTOP-1)/4.0)        ;first template block to be read
        SB=INT(FLOAT(CTOP(II)-1)/4.0)    ;first scene block
        N1=MOD((TTOP-1),4)
        N2=MOD((CTOP(II)-1),4)
        TLS=TLEFT
        SLS=SLEFT(II)
C
C       User check of window parameters
C
        IF(SUPPRESS)GO TO 45
        IF(N.EQ.1)GO TO 45
        TYPE"<15>","********************************************
     $**********************"
        TYPE"<15>","<11><11>******* WINDOW",(II-10),"  *******"
        WRITE(10,2000)INFILE1(1)
 2000   FORMAT(//,10X,"Template file name ---> ",S13)
        WRITE(10,3000)INFILE2(1)
 3000   FORMAT(10X,"   Scene file name ---> ",S13,/)
        TYPE"<15>","<11><11>WIDTH=",WIDTH,"<11>   LENGTH=",LENGTH
        TYPE"     TEMPLATE TOP ROW=",TTOP,"<11>   SCENE TOP ROW=",CTOP(I)
        TYPE" TEMPLATE LEFT COLUMN=",TLS,"<11>   SCENE LEFT COLUMN
     $=",SLS,"<15>"
C
C       Begin evaluating the energy (or distances)
C
 45     CORREL=0.0          ;initialize distances
        NL1DIST=0.0
        CALL XRDBLK(1,TB,TEMP,1,IER)
        IF(IER.NE.1)TYPE"1RDBLK #",TB," error:",IER
        CALL XRDBLK(2,SB,SCENE,1,IER)
        IF(IER.NE.1)TYPE"2RDBLK #",SB," error:",IER
C
C       This module will continue to loop until the
C       search windows have been completed (i.e. # of
C       iterations=(length of the window)/4)
C
 100    CALL EUCLID(N1,N2,N,$500)
 110            TB=TB+1
                SB=SB+1
                IF(N1.EQ.0)CALL XRDBLK(1,TB,TEMP,1,IER)
                IF(IER.NE.1)TYPE"1RDBLK #",TB," error:",IER
                IF(N2.EQ.0)CALL XRDBLK(2,SB,SCENE,1,IER)
                IF(IER.NE.1)TYPE"2RDBLK #",SB," error:",IER
                CALL EUCLID(N1,N2,N,$500)
                IF(N1.EQ.N2)GO TO 110
                IF(N1.EQ.0)CALL XRDBLK(1,TB,TEMP,1,IER)
                IF(IER.NE.1)TYPE"1RDBLK #",TB," error:",IER
                IF(N2.EQ.0)CALL XRDBLK(2,SB,SCENE,1,IER)
                IF(IER.NE.1)TYPE"2RDBLK #",SB," error:",IER
        GO TO 100
C
C       Store the distances computed for iteration II (or
C       temporarily store the energies if N=1)
C
 500    IF(SENERGY.LT.1.0)SENERGY=1.0
        IF(TENERGY.LT.1.0)TENERGY=1.0
        IF(SL1ENERGY.LT.1.0)SL1ENERGY=1.0
        IF(TL1ENERGY.LT.1.0)TL1ENERGY=1.0
        S2NORM=SQRT(SENERGY)
        T2NORM=SQRT(TENERGY)
        S1NORM=SL1ENERGY
        T1NORM=TL1ENERGY
        AREA=FLOAT(LENGTH)*FLOAT(WIDTH)
        IF(N.EQ.1)GO TO 40
C
C       Compute distance factors
C
        NDISTL1(II)=NL1DIST
        NXYMIN=TL1ENERGY/(SQRT(AREA*TENERGY))
        NL2MAX=SQRT(2.0*(1.0-NXYMIN))
        CORPEAK(II)=CORREL/(S2NORM*T2NORM)
        CFACTOR(II)=INT(100.0*(CORPEAK(II)-NXYMIN)/(1.0-NXYMIN)+0.5)
        IF(CFACTOR(II).LE.0)CFACTOR(II)=0
        NL2=SQRT(2.0*(1.0-CORPEAK(II)))
        L2FACTOR(II)=INT(100.0*(1.0-NL2/NL2MAX)+0.5)
        IF(L2FACTOR(II).LE.0)L2FACTOR(II)=0
        L1FACTOR(II)=INT(100.0*(1.0-NDISTL1(II)/NL1MAX)+0.5)
        IF(L1FACTOR(II).LE.0)L1FACTOR(II)=0
C
C       Reset the template window parameters
C
                TTOP=ITOP
                TLEFT=ILEFT
                WIDTH=IWIDTH
                LENGTH=ILENGTH
 600    CONTINUE
        TYPE"<7><7>*********************************"
C
C******** WRITE RESULTS TO LINEPRINTER *************************
C
C
C       User input of comment
C
        DO 625 I=1,200
                COMMENT(I)=0
 625    CONTINUE
        ACCEPT"Enter a 1 to add comment to the output: ",IOPT
        IF(IOPT.NE.1)GO TO 650
        ACCEPT"Enter # of comment lines (max=4): ",NUMCOM
        IF(NUMCOM.LT.1)NUMCOM=1
        NUMCOM=MIN(NUMCOM,4)
        DO 640 I=1,NUMCOM
```

```
                TYPE"Enter comment line #",I," to be printed with
     $ results between the arrows:"
                TYPE"|<11><11><11><11><11><11><11><11><11>|"
                TYPE"V<11><11><11><11><11><11><11><11><11>V"
                READ(11,9999)COMMENT((50*(I-1)+1))
 640    CONTINUE
C
C       Write output header
C
 650    WRITE(12,9005)
        WRITE(12,9000)
        WRITE(12,8000)
        IF(REDUCED)WRITE(12,7500)
        WRITE(12,7000)LENGTH,TTOP;,TL1ENERGY
        WRITE(12,6000)INFILE1(1),WIDTH,TLS;,TENERGY
        IF(REDUCED)WRITE(12,5500)INFILE2(1)
        IF(.NOT.REDUCED)WRITE(12,5000)INFILE2(1)
        WRITE(12,4600)
        WRITE(12,4500)
        WRITE(12,4200)
        WRITE(12,4100)
C
C       Write distance factors
C
        II=10
        DO 710 JJ=1,NUMPEAKS
        DO 700 KK=1,9
        II=II+1
        A=CFACTOR(II)
        B=L2FACTOR(II)
        C=L1FACTOR(II)
        SCORE=INT((A*B*C)**(1.0/3.0)+0.5)
        IF(KK.NE.1)GO TO 698
        WRITE(12,4000)JJ,PROW(JJ),PCOLUMN(JJ),ROWCENT(II),COLCENT(II),
     $CTOP(II),SLEFT(II),CFACTOR(II),L2FACTOR(II),L1FACTOR(II),
     $SCORE
 4000   FORMAT(14X,I2,":",I4,",",I3,5X,I3,",",I3,6X,I3,4X,I3,
     $7X,I3,6X,I3,5X,I3,5X,I3,T132," ")
        GO TO 700
 698    WRITE(12,4050)ROWCENT(II),COLCENT(II),CTOP(II),SLEFT(II),
     $CFACTOR(II),L2FACTOR(II),L1FACTOR(II),SCORE
 4050   FORMAT(30X,I3,",",I3,6X,I3,4X,I3,
     $7X,I3,6X,I3,5X,I3,5X,I3,T132," ")
 700    CONTINUE
        WRITE(12,4051)
 4051   FORMAT(" ")
 710    CONTINUE
C
C       Write comments to lineprinter
C
        IF(IOPT.NE.1)GO TO 800
        DO 790 I=1,NUMCOM
                WRITE(12,9500)COMMENT((50*(I-1)+1))
 790    CONTINUE
 800    WRITE(12,9000)
C
C       Format statements
C
 9999   FORMAT(S100)
 9500   FORMAT(16X,"COMMENT:  ",S100)
 9005   FORMAT(///////////)
 9000   FORMAT(/,11X,81("*"),T132," ")
 8000   FORMAT(///,37X,"RECOGNITION RESULTS <10>",///)
 7500   FORMAT(15X," ****REDUCED****")
 7000   FORMAT(15X," TEMPLATE WINDOW:",7X,"LENGTH=",I3," ROWS",12X,"TOP
     $I3,10X,T132," ")
 6000   FORMAT(16X," (",S13,")",8X,"WIDTH=",I3," COLUMNS",9X,"LEFTCOL=",
     $I3,10X,T132," ")
 5500   FORMAT(//,30X,"*REDUCED* SCENE FILE --->  ",S13,T132," ",//)
 5000   FORMAT(//,35X,"SCENE FILE --->  ",S13,T132," ",//)
 4600   FORMAT(14X," CORRELATION     WINDOW")
 4500   FORMAT(14X,"    PEAK            CENTER      TOP    LEFT
     $  CORRELATE   L2      L1")
 4200   FORMAT(14X,"(ROW,COLUMN)  (ROW,COLUMN)   ROW   COLUMN
     $   FACTOR   FACTOR  FACTOR  SCORE")
 4100   FORMAT(14X,"-----------  ------------   ---   ------
     $  --------  ------  ------  -----")
        TYPE"<7><15><11>**** CHECK LINEPRINTER FOR RESULTS *****<15>"
C
C****** Present Option Menu ********************************
C
        GO TO 2010
 2002   TYPE"<15>","Input error.<7>  Try again."
 2010   TYPE"<15>","******************************************
     $************************"
        TYPE"<15>","What next?<15>","Here are the options:"
        TYPE"<15>","<11>1 - Try another set of windows"
        TYPE"<15>","<11>2 - Start over with new input pictures"
        TYPE"<15>","<11>3 - STOP<15>"
        ACCEPT"<11>Enter option ---> ",IOPT
        IF(IOPT.LT.1.OR.IOPT.GT.3)GO TO 2002
        TYPE"<15>"
        IF(IOPT.EQ.1)GO TO 3
        CALL RESET
        IF(IOPT.EQ.2)GO TO 99
        STOP
        END
C
C********* Program DISTANCE *******************************************
```

```
      SUBROUTINE EUCLID(TSTART,SSTART,N,$)
C***********************************************************************
C
C     (Called by DISTANCE)              by Lt Jim Cromer
C
C     If N=1 ---> calculate L1 and L2 energies of
C                 template and scene windows
C     Else   ---> calculate the normalized L1 and cross-
C                 correlation measures between the windows
C
C     TSTART, SSTART are the row position within the packed
C     video block of the first row of the window.  They
C     are automatically incremented after the first call to
C     EUCLID (TSTART, SSTART between 0-3 inclusive).
C
C***********************************************************************
      REAL NL1DIST,L1DIST,NSUMSQ
      INTEGER SCENE(1024),TEMP(1024),SLS,TLS,SSTART,TSTART,WIDTH
      COMMON /LIST1/ SCENE,TEMP,SLS,TLS,L1DIST,NL1DIST
      COMMON /LIST2/ LENGTH,WIDTH,IROWCOUNT,SUMSQ,NSUMSQ
      COMMON /LIST3/ SL1ENERGY,TL1ENERGY,SENERGY,TENERGY
      COMMON /LIST4/ S1NORM,S2NORM,T1NORM,T2NORM,CORREL
C
C     Set do loop parameters
C
      JMIN=MAX(SSTART,TSTART)
      DO 2 J=JMIN,3
      K=TSTART*256+TLS
      M=SSTART*256+SLS
      KMAX=K+WIDTH-1
      IF(N.EQ.1)GO TO 3
C
C     Calculate the distances
C
      DO 1 L=K,KMAX
            RSCENE=FLOAT(SCENE(M))
            RTEMP=FLOAT(TEMP(L))
            NL1DIST=ABS((RSCENE/S1NORM)-(RTEMP/T1NORM))+NL1DIST
            CORREL=(RSCENE*RTEMP)+CORREL
            M=M+1
 1    CONTINUE
      GO TO 5
C
C     Calculate the energies
C
 3    DO 4 L=K,KMAX
            RSCENE=FLOAT(SCENE(M))
            RTEMP=FLOAT(TEMP(L))
            TL1ENERGY=TL1ENERGY+RTEMP
            SL1ENERGY=SL1ENERGY+RSCENE
            SENERGY=(RSCENE**2)+SENERGY
            TENERGY=(RTEMP**2)+TENERGY
            M=M+1
 4    CONTINUE
```

```
C
C       Test for the end of the window
C
 5      IROWCOUNT=IROWCOUNT+1
        IF(IROWCOUNT.GE.LENGTH)RETURN 4
C
C       Increment block row counters
C
        TSTART=TSTART+1
  2     SSTART=SSTART+1
        IF(SSTART.EQ.4)SSTART=0
        IF(TSTART.EQ.4)TSTART=0
        RETURN
        END
C
C******** Subroutine EUCLID *********************************************
```

APPENDIX G:

SUPPORT SUBROUTINES

This appendix contains the following programs:

1. IOF

2. TIMER

3. UNPACK

4. XRDBLK

```
        SUBROUTINE IOF(N,MAIN,F1,F2,F3,MS,S1,S2,S3)
C****************************************************************
C
C       Written by Lt. Simmons          10 Sep 1981
C       Version 2
C
C       This FORTRAN 5 subroutine will read from the file
C       COM.CM (FCOM.CM in the foreground) the program name,
C       any global switches, and up to three local file
C       names and corresponding local switches.
C
C       Calling arguments:
C
C       N is the number of local files and switches to be
C       read from (F)COM.CM.  N must be 1, 2, or 3.
C
C       MAIN is an ASCII array for the main program file name.
C
C       F1, F2, and F3 are the three ASCII arrays to return
C       the local file names.
C
C       MS is a two-word integer array that holds any global
C       switches.
C
C       S1, S2, and S3 are two-word integer arrays that
C       hold the local switches corresponding to F1 through
C       F3 respectively.
C
C******************************************************************
C
C       Dimension the arrays.
C
        DIMENSION MAIN(7),MS(2)
        INTEGER F1(7),F2(7),F3(7),S1(2),S2(2),S3(2)
C
C       Check the bounds on N.
C
        IF(N.LT.1.OR.N.GT.3)STOP "N out of bounds in IOF."
C
C       Process the data in (F)COM.CM
C
        CALL GROUND(I)  ;Find out which ground program is in
        IF(I.EQ.0)OPEN 0,"COM.CM"       ;Open ch. 0 to COM.CM
        IF(I.EQ.1)OPEN 0,"FCOM.CM"      ;Open ch. 0 to FCOM.CM
        CALL COMARG(0,MAIN,MS,IER)      ;Read from (F)COM.CM
        IF(IER.NE.1)TYPE" COMARG error:",IER
        WRITE(10,1)MAIN(1)              ;Type program name
   1    FORMAT(' Program ',S13,'running.')
        CALL COMARG(0,F1,S1,JER)        ;Read from (F)COM.CM
        IF(JER.NE.1)TYPE" COMARG error (F1):",JER
        IF(N.EQ.1)GO TO 2               ;Test N
        CALL COMARG(0,F2,S2,KER)        ;Read from (F)COM.CM
        IF(KER.NE.1)TYPE" COMARG error (F2):",KER
        IF(N.EQ.2)GO TO 2               ;Test N
```

```
      CALL COMARG(0,F3,S3,LER)        ;Read from (F)COM.CM
      IF(LER.NE.1)TYPE" COMARG error (F3):",LER
  2   CLOSE 0
      RETURN
      END
C
C************ Subroutine IOF ***************************************
```

```
      SUBROUTINE TIMER(I)
C*********************************************************************
C
C     Subroutine TIMER       Written by  Lt. Jim Cromer
C     Fortran 5
C
C     This subroutine is used to time the real-time execution
C     time of the calling program.  If the parameter passed, I,
C     is equal to 0, the timer is unconditionally started.
C     If I is not equal to 0, the timer is unconditionally
C     stopped, and the total run time is typed on the console
C     CRT.
C
C     Execution Line Format
C             CALL TIMER(I)     ;IF(I.EQ.0), start timing
C                               ;IF(I.NE.0), stop timing
C
C**********************************************************************
      COMMON /ITIME/ IH1,IM1,IS1
      IF(I.NE.0)GO TO 100
      CALL FGTIME(IH1,IM1,IS1)        ;get starting time
      WRITE(10,1000)IH1,IM1,IS1
 1000 FORMAT(//" START TIME --->",I4,":",I3,":",I3)
      RETURN
  100 CALL FGTIME(IH2,IM2,IS2)        ;get stopping time
      WRITE(10,2000)IH2,IM2,IS2
 2000 FORMAT(//"  STOP TIME --->",I4,":",I3,":",I3)
      ITOTAL=3600*(IH2-IH1)+60*(IM2-IM1)+IS2-IS1
      HOURS=INT(ITOTAL/3600)
      TRON=(ITOTAL-3600*HOURS)        ;intermediate variable
      MINS=INT(TRON/60)
      ISECS=MOD(TRON,60)
      WRITE(10,3000)HOURS,MINS,ISECS
 3000 FORMAT(//" TOTAL TIME --->",I4,":",I3,":",I3)
      RETURN
      END
C
C************** Subroutine TIMER ***********************************
```

```
C****************************************************************
C
C       Unpacking (packing) routines
C       Written by Lt. Simmons          Version 2
C       Documented by Lt. Cromer
C
C       These subroutines unpack (repack) four 4-bit integers from a
C       16-bit integer word.  The pixels in a video file have to
C       be unpacked if each pixel is to be operated on separately.
C
C       Packed video (4 pixels/1 word):
C                     - - - - - - - - - - - - - - - -
C        WORD(N+1)  |PIXEL 1|PIXEL 2|PIXEL 3|PIXEL 4|
C                     - - - - - - - - - - - - - - - -
C
C       Unpacked video (4 pixels/4 words):
C                     - - - - - - - - - - - - - - - -
C        WORD(X  )  |   <- unused ->         |PIXEL 1|
C                     - - - - - - - - - - - - - - - -
C                     - - - - - - - - - - - - - - - -
C        WORD(X+1)  |   <- unused ->         |PIXEL 2|
C                     - - - - - - - - - - - - - - - -
C                     - - - - - - - - - - - - - - - -
C        WORD(X+2)  |   <- unused ->         |PIXEL 3|
C                     - - - - - - - - - - - - - - - -
C                     - - - - - - - - - - - - - - - -
C        WORD(X+3)  |   <- unused ->         |PIXEL 4|
C                     - - - - - - - - - - - - - - - -
C
C               where N=X mod 4
C
C******************************************************************
        SUBROUTINE UNPACK(N,PIXWORD,PIXELS)
        INTEGER PIXWORD(N),PIXELS(4,N)    ;Four pixels per word
        DO 1 I=1,N                        ;'N' allows higher-order
        DO 1 J=1,4                        ;arrays to be passed.
        PIXELS((5-J),I)=15.AND.PIXWORD(I) ;Pick off right pixel
   1    PIXWORD(I)=ISHFT(PIXWORD(I),-4)   ;Shift word 4 bits right
        RETURN                            ;to pick off next pixel.
        END
C
        SUBROUTINE REPACK(N,PIXELS,PXWD)
        INTEGER PIXELS(4,N),PXWD(N)
        DO 1 J=1,N
        PXWD(J)=0
         DO 1 I=1,4
         PXWD(J)=ISHFT(PXWD(J),4)
  1      PXWD(J)=PIXELS(I,J)+PXWD(J)
        RETURN
        END
C
C************ Packing Subroutines *********************************
```

```
        SUBROUTINE XRDBLK(CH,J,FILE,I,IER)
C***********************************************************************
C
C       by Lt. Jim Cromer
C       Subroutine XRDBLK performs a RDBLK to the designated
C       channel, reads a packed video file block, and
C       returns an unpacked array.
C
C***********************************************************************
        INTEGER CH,FILE(1024),VIDEO(256)
        K=256*I
        IF(J.GE.0.AND.J.LE.63)GO TO 1
        TYPE"ERROR: <7>BLOCK POINTER OUT OF BOUNDS IN XRDBLK"
        TYPE"           J=",J
        STOP
 1      IF(I.EQ.1)GO TO 2
        TYPE"ERROR IN <7>XRDBLK"
        TYPE" # Blocks to be read =",I
        STOP
 2      CALL RDBLK(CH,J,VIDEO,I,IER)
        DO 3 L=1,K
        DO 3 M=1,4
                ICOUNT=5-M+(L-1)*4
                FILE(ICOUNT)=15.AND.VIDEO(L)
                VIDEO(L)=ISHFT(VIDEO(L),-4)
 3      CONTINUE
        RETURN
        END
C
C********* Subroutine XRDBLK ***********************************
```

APPENDIX H: PRELIMINARY RESULTS

Summary of Tank SCOREs

Template and Tank windows both globally normalized

Template	Scene	High	Low	Average
PTEMPH3	PTANKH3	35	0	19
RPTEMPH3	RPTANKH3	36	0	5
PTEMPD4	PTANKB2	41	0	25
PTEMPD4	PTANKC3	49	0	15
PTEMPD4	PTANKE2	56	0	31
PTEMPD4	PTANKG4	0	0	0
	Average	36	0	16

Template and Tank windows both grid normalized (9x5 grid used)

Template	Tank	High	Low	Average
NORMD4	NORMB2	65	40	52
''	NORMC3	69	43	58
''	NORMD2	64	43	54
''	NORME2	67	45	57
''	NORMG4	42	16	29
''	NORMH3	79	24	43
	Average	64	35	49

Summary of Scene SCOREs

Template and Scene windows both globally normalized

Template	Scene	High	Low	Average
PTEMPH3	PSCENEI3	16	14	15
PTEMPH3	PSCENEO4	24	18	21
RPTEMPH3	RPSCENEO4	20	10	15
RPTEMPH3	RPSCENEL1	19	14	16
	Average	19	14	17

Template and Scene windows both grid normalized (9x5)

Template	Scene	High	Low	Average
NORMD4	WHITE	0	0	0
NORMD4	NORM7	15	2	8

VITA

James H. Cromer was born on 10 April 1959 in Cleveland, Ohio. He graduated from Kenston High School, Chagrin Falls, Ohio, in 1977. He attended Grove City College, Grove City, Pennsylvania, on a 4-year AFROTC scholarship. He received the Bachelor of Science degree in Electrical Engineering, and was commissioned a Second Lieutenant in the United States Air Force, in May 1981. He entered the School of Engineering, Air Force Institute of Technology, in June 1981, and was chosen to receive the 1981-1982 IEEE Outstanding Student Award in May 1982. He is a member of Tau Beta Pi, Sigma Pi Sigma, and Eta Kappa Nu.

Permanent Address: 204 E. Mill St.
Port Allegany, PA 16743

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GE/EE/82D-26	2. GOVT ACCESSION NO. AD-A124788	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SCENE ANALYSIS: NON-LINEAR SPATIAL FILTERING FOR AUTOMATIC TARGET DETECTION		5. TYPE OF REPORT & PERIOD COVERED MS Thesis
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) James H. Cromer 2nd Lt USAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE December 1982
		13. NUMBER OF PAGES 168
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release;
distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Approved for public release: IAW AFR 190-17.
Lynn Wolaver
LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB OH 45433

4 JAN 1982

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Cross-correlation	Pattern Recognition	Fortran
Form Analysis	Scene Analysis	
Image Processing	Target Classification	
Energy Normalization	Target Detection	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This work focuses on a method for two-dimensional pattern recognition. The method includes a global search scheme for candidate windows of interest, based on Fourier domain cross-correlation. A method to normalize the input scene by local rectangular regions, in an attempt to efficiently approximate search window normalization, is presented. Also developed is a candidate window (potential target) similarity measure, based on the normalized L1 and Euclidean distances, which is independent of the template DC value and its energy. Observations on the performance of the algorithm applied to visual spectrum photographs of tanks in a realistic environment are included. Also included is the software needed to implement the algorithm on Data General Eclipse S/250 minicomputer.

3-8

DT